AFRL-RH-WP-TR-2012-0005

TENTACLE: Multi-Camera Immersive Surveillance System

Justin W. Benjamin
Benjamin L. Post
Kyle K. Estes
Randy L. Milbert

Primordial, Inc.
1021 Bandana Boulevard East
Saint Paul MN 55108

DECEMBER 2011
FINAL REPORT

THIS IS A SMALL BUSINESS INNOVATIVE RESEARCH (SBIR) PHASE I REPORT.

AIR FORCE RESEARCH LABORATORY
711 HUMAN PERFORMANCE WING,
HUMAN EFFECTIVENESS DIRECTORATE,
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433
AIR FORCE MATERIEL COMMAND
UNITED STATES AIR FORCE

NOTICE AND SIGNATURE PAGE

AFRL-RH-WP-TR-2012- 0005 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

\\signed\\
DARREL G. HOPPER
Work Unit Manager
Battlepace Visualization Branch

\\signed\\
JEFFREY L. CRAIG
Chief, Battlespace Visualization Branch
Decision Making Division

\\signed\\
MICHAEL A. STROPKI
Chief, Decision Making Division
Human Effectiveness Directorate
711 Human Performance Wing

PRIMORDIAL

PRIMORDIAL.COM

MEMORANDUM FOR RECORD

FROM: Primordial, Inc., 1021 Bandana Blvd East Ste 225, Saint Paul MN 55108

SUBJECT: Agreement and Approval for Public Release of SBIR Data Rights (FA8650-11-M-6193)

1. Primodial, Inc. waives its SBIR Data Rights for the attached final report entitled "TENTACLE: Multi-Camera Immersive Surveillance System" for SBIR Phase I contract FA8650-11-M-6193. The Government is granted an unlimited nonexclusive license to use, modify, reproduce, release, perform, display or disclose this final report and the data contained therein.

2. The attached has been reviewed and is approved for public release, distribution unlimited. The following request is published in the interest of scientific and technical information exchange and does not constitute Government approval or disapproval of the ideas or findings.

Randy L. Milbert

Randy L. Milbert, President
Primordial, Inc.

(Date) 5 Dec 2011

[illegible]

Principal Electronics Engineer
711 HPW/RHCV

(Date) 5 Dec 2011

[illegible]

(Branch Chief Title/Signature)

(Date) 14 Dec 2011

[illegible]

STINFO Title/PA Submitter /Signature)

(Date) 27 Dec 11

1.0 REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE ***(DD-MM-YY)*** 5-12-2011	**2. REPORT TYPE** Final	**3. DATES COVERED** ***(From - To)*** 7 Mar 2011 - 5 Dec 2011

4. TITLE AND SUBTITLE TENTACLE™: Multi-Camera Immersive Surveillance System	**5a. CONTRACT NUMBER** FA8650-11-M-6193
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65505F
6. AUTHOR(S) Justin W. Benjamin, Benjamin L. Post, Kyle K. Estes, and Randy L. Milbert	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER CV
	5f. WORK UNIT NUMBER 3005CV09
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Primordial, Inc. 1021 Bandana Boulevard East Saint Paul MN 55108	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Materiel Command Air Force Research Laboratory 711Human Performance Wing, Human Effectiveness Directorate Decision Making Division, Battlespace Visualization Branch Wright-Patterson AFB OH 45433-7022	**10. SPONSORING/MONITORING AGENCY ACRONYM(S)** AFRL/RHCV
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S) **AFRL-RH-WP-RH-2012-0005**

12. DISTRIBUTION/AVAILABILITY STATEMENT:

Distribution A: Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES:

88ABW/PA Cleared 02-16-12; 88ABW-2012-0850

14. ABSTRACT: This final technical report describes the results of collaborative efforts by Primordial, intuVision, and All Hazards Management in conducting a phase I Small Business Innovation Research (SBIR) effort, titled AF103-032: Multi-camera real-time Feature Recognition, Extraction &Tagging Automation (McFRETA), and authored by the U.S. Air Force Research Laboratory (AFRL). The goal of the effort was to design, develop, and demonstrate a scalable framework and software application to enable tracking and feature extraction of uncooperative entities capture in multiple streaming sources and enabling near real-time forensic analysis of the extracted data. During the effort, Primordial worked with intuVision to identify key features to extract from live video feeds to include geographic location, entity type, and dominate colors of the entity. Primordial then built a graphical user interface to display avatars representing real world entities in a three dimensional virtual reality using the WorldWind SDK. To prove feasibility of the system, Primordial conducted 13 internal demonstrations to validate the results of the system.

15. SUBJECT TERMS

Real-time video surveillance, people and vehicle tracking, multi-camera multi-sensor fusion algorithms, automated Identification and tagging, animated graphical interface, metadata, extraction, framework, database, query, event

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT:**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** (Monitor)
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	SAR	78	Dr. Darrel G. Hopper **19b. TELEPHONE NUMBER** *(Include Area Code)* (937) 255-8822

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18

This page intentionally left blank.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

GLOSSARY

2D	Two-dimensional
3D	Three-dimensional
Aforge.net	Open source software used in computer vision applications and image processing.
AHM	All Hazards Management, Inc., a subcontractor to Primordial on the project.
API	Application Programming Interface, a particular set of rules and specifications that software programs can follow to communicate with each other.
AVI	Audio Video Interleave, a format for video files
BFT	A TDL used by the Army
Bosch	Manufacturer of the Divar Control Center software which is installed at Camp Ripley
BQS	Boolean Query Syntax
C2	Command and control
CCIRM	Collection Coordination and Information Requirements Management, a metadata layer in STANAG 4559
CCTV	Closed Circuit Television
CoT	Cursor on Target, an XML schema for exchanging time-sensitive positions of moving objects between systems.
CPU	Central Processing Unit
C-Thru	Software developed by AHM to represent a real world environment in a virtual 3D environment.
CUDA	A parallel computing architecture designed by Nvidia Corporation that runs on GPUs.
FBCB2	Force XXI Battle Command, Brigade and Below, a United States Army system for C2 of tactical vehicles
FOV	Field of view
FPS	Frames per second, a measurement of how quickly video frames are being transmitted.
FST21	An Israeli company that manufactures the SafeRise system, which uses video cameras and facial recognition to provide secure entry control to a facility
GHz	Gigahertz, a measurement of CPU speed.
GMTI	Ground Moving Target Indicator, a system that tracks moving objects via SAR
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HSV	Hue, Saturation, Value
intuVision	intuVision, Inc., a subcontractor to Primordial on the project, and experts in object detection and tracking algorithms.
intuVision Panoptes SDK	An SDK allowing third party developers to integrate intuVision's Panoptes software capabilities into their own
IP	Internet protocol

IPL	ISR Product Library, or Image Product Library, or Intelligence Product Library
ISR	Intelligence, Surveillance, and Reconnaissance
ISR product	An image, video, or other piece of data relevant to ISR operations
JPEG	Joint Photographic Experts Group, a commonly used method of lossy compression for digital images.
LAN	Local Area Network
Link 16	A TDL used by the Air Force
LINQ	Language Integrated Query, a Microsoft .NET technology for standard query operations.
Microsoft .NET Framework	A software framework for developing applications for Windows
MIL-STD-2525B	Military Standard 2525-B, a standard for military symbology.
MJPEG	Motion JPEG, a method of compressing video frames as separate JPEG images.
MTS	Movement Tracking System, a United States Army system for C2 of logistical vehicles
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organization
NSIL	NATO Standard ISR Library
NSILI	NATO Standard ISR Library Interface
Nvidia Ion	A computer architecture designed by Nvidia Corporation that employs a Nvidia-made GPU and an Intel Atom CPU.
OpenCV	Open Computer Vision, an open source software kit used in computer vision applications and image processing.
Panoptes	Software developed by intuVision which performs video analysis.
PC	Personal computer
Primordial	Primordial, Inc., the prime contractor on the project.
QVGA	Quarter Video Graphics Array, a graphics display resolution of 320 x 240 pixels.
RAM	Random access memory
RGB	An additive color model in which red, green, and blue light is added together in various ways to reproduce a broad array of colors.
RSS	Really Simple Syndication, a standard format by which blog entries, news headlines, audio, and video is disseminated via the web.
SAR	Synthetic Aperture Radar
SBIR	Small Business Innovation Research
SDK	Software Development Kit, a set of development tools that allows for the creation of applications with a particular capability.
SQL	Structured Query Language, a standard language for accessing databases.
STANAG	NATO Standardization Agreement
STANAG 4559	NATO Standard ISR Library Interface
T2DL	Tentacle TDL
TC	Tentacle Client
TCP/IP	Transmission Control Protocol/Internet Protocol

TDL	Tactical Data Link, a stream of WWW information about battlefield entities
Tentacle	Software developed by Primordial to facilitate information passing between Panoptes and C-Thru.
TIGR	Tactical Ground Reporting System, a web-based information sharing system available to the United States Army
TIPL	Tentacle IPL
TM	Tentacle Mote
TS	Tentacle Server
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
VGA	Video Graphics Array, a graphics display resolution of 640 x 480 pixels.
VMTI	Video Moving Target Indicator, a system that tracks moving objects via video cameras
WGS84	World Geodetic System 1984
WWW information	What, where, and when
XML	Extensible Markup Language

FOREWORD

This PE65502F $99,221.00 SBIR Phase I contract FA8650-11-M-6193 (AFRL WU 3005CV09 was awarded to Primordial, Inc 7 March 2011 with an end date of 5 December 2011. The Primordial contract was one of four Phase I efforts awarded under SBIR BAA AF10.3 Topic AF093-032 entitled "McFRETA: Multi-camera real-time Feature Recognition, Extraction & Tagging Automation."

The objective of the McFRETA program is to develop an open and scalable framework/tool to perform automated feature recognition of multiple streaming sources and to extract metadata and make available for both ongoing operations and forensics. Sub-objectives were to (a) identify algorithms for feature recognition and extraction suitable for real time application,(b) identify suitable metadata tags that allow for human and machine devices search criteria, (c) devise a framework that would function in orchestration and event processing frameworks, and (d) design a prototype system. The ultimate objective is automated identification, tagging, and tracking of humans and vehicles from multiple real time video feeds. A framework and demonstration of how existing and new algorithms can be incorporated and tested is also sought, and of how an operator can develop queries and rules that assist assessment and execution. Scalability from tactical to regional areas of interest is required. The tool sought comprises definition of an open framework for integration of real-time feature recognition and extraction algorithms, generation of a stream of standardized metadata associated with the content source, and design and demonstration of an open, scalable system that supports queries and event/alert notification based on rule sets. Due to the heterogeneous nature of the content capture and storage systems as well as the operations (or forensics) systems, the integrating framework must be open and user friendly so as to enable queries in a broad manner.

1.0 INTRODUCTION

As the amount of information being received from sensors in the battlefield increases, the need for effective processing of data grows drastically. Important events and information must be extracted from the flood of sensor data in a meaningful way. Battlefield leaders must be able to quickly evaluate this data and use it to make the correct decisions. As the amount of data grows, this becomes impossible for a human being without some sort of automated assistance. Computer systems are needed to automatically flag important information in sensor data and categorize it, as well as allow leaders to search for events and activities relevant to their decision-making.

Tentacle achieves this goal by automatically extracting information about entities in live video streams, flagging the entities with metadata such as color, entity type and geographic location, and representing the entities and their metadata in a single 3D rectified environment as shown in Figure 1. Tentacle also allows real-time querying of extracted information and a system for alerting the user if a tracked entity breaks a given rule, such as entering a restricted area. Using these techniques, Tentacle has the potential to reduce the cognitive load for the battlefield decision maker and make processing incoming data more efficient and effective.

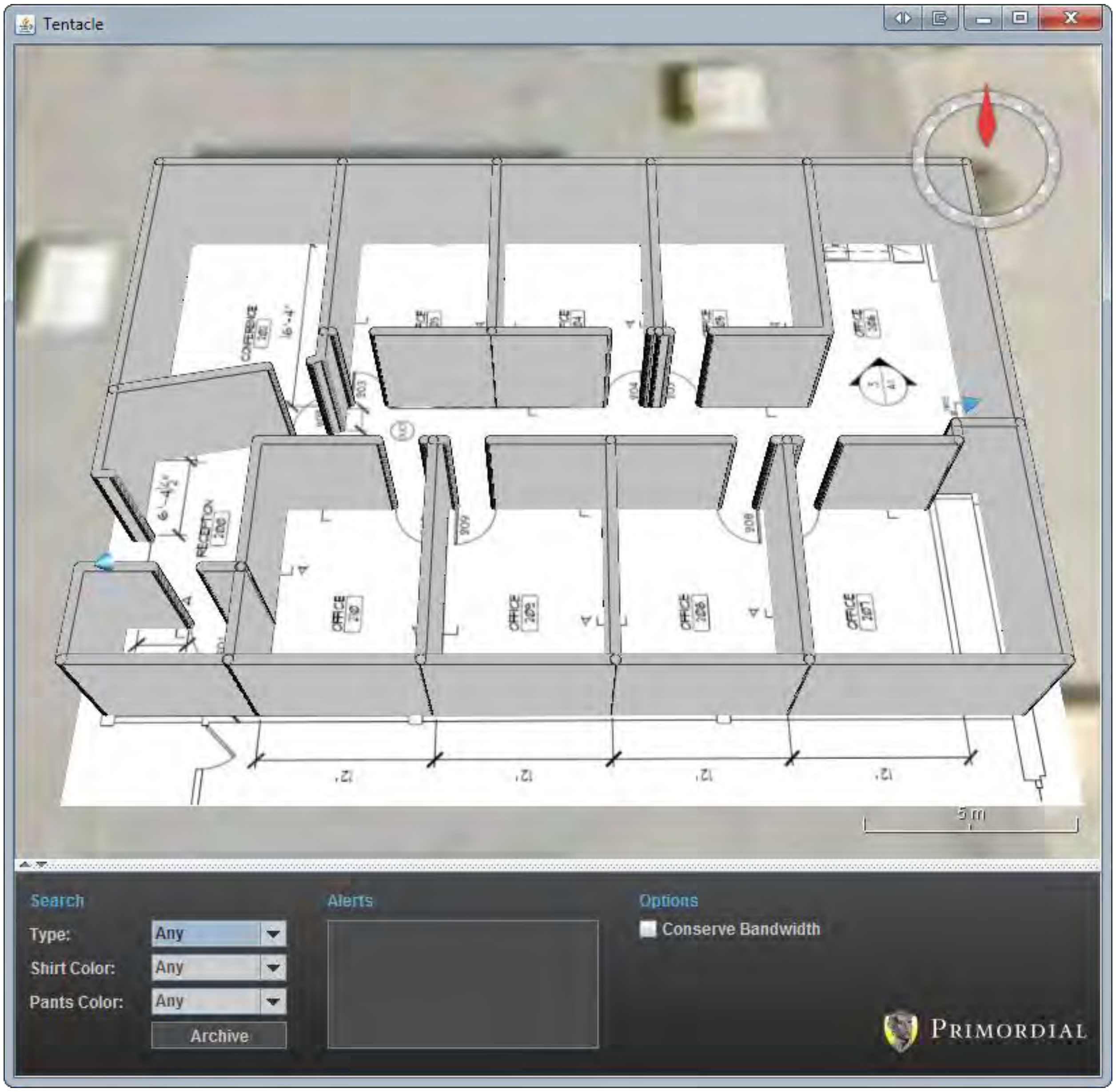


Figure 1: Tentacle Client GUI

Section 2.0 contains a detailed discussion of the results we obtained in phase I; however, here is a short summary of these results.

1.1 Target Tracking

Primordial, both with the help of its partner intuVision as well as independently, developed methods for detecting and tracking entities pictured in a camera stream. The intuVision and Primordial methods both use an OpenCV-based approach to processing the incoming video frames and extract features. Both the intuVision-based and the in-house systems were proven effective during ongoing testing.

1.2 2D to 3D Projection Algorithm

A 2D-to-3D projection algorithm was developed to accurately correlate pixel locations in a video frame with real-world geographic coordinates. This formula uses a vector to ground-plane intersection method for finding a tracked entity's location in 3D space. The algorithm was proven accurate through numerous ongoing tests during the development of Tentacle.

1.3 Server

Primordial created a simple server system for processing data from motes to clients and to allow for later implementation of an archival system. The server reliably and accurately transmits tracked entity and alert information from multiple sensor motes to the client interfaces.

1.4 Client

We created a graphical client interface based on NASA WorldWind in Java. The client interface displays entities being tracked by multiple sensor motes in near real-time and allows the user to filter entities on the screen by various criteria such as primary and secondary color and entity type. The client also displays alerts detected by the motes using rules predefined by the end user such as tripwires and off-limit locations.

2.0 DISCUSSION OF RESULTS

2.1 Securing Partnerships

Early in the project, Primordial worked to establish subcontracts with the two companies with whom we proposed to work.

2.1.1 Panoptic/All Hazards Management

Primordial entered into an agreement with All Hazards Management (AHM) and Panoptic for phase I of this SBIR effort. AHM and Panoptic were tasked with providing consultation and support for building the end user graphical interface for Tentacle.

2.1.2 intuVision

Primordial entered into an agreement with intuVision, Inc. intuVision was tasked with providing support for the demonstration of the Tentacle system by recommending a hardware platform and providing technical assistance as needed. Additionally, intuVision was tasked with providing support for the Tentacle project by supplying a copy of its Panoptes software as well as modifications to Panoptes to support Tentacle. The administration GUI of intuVision's Panoptes product is depicted in Figure 2.

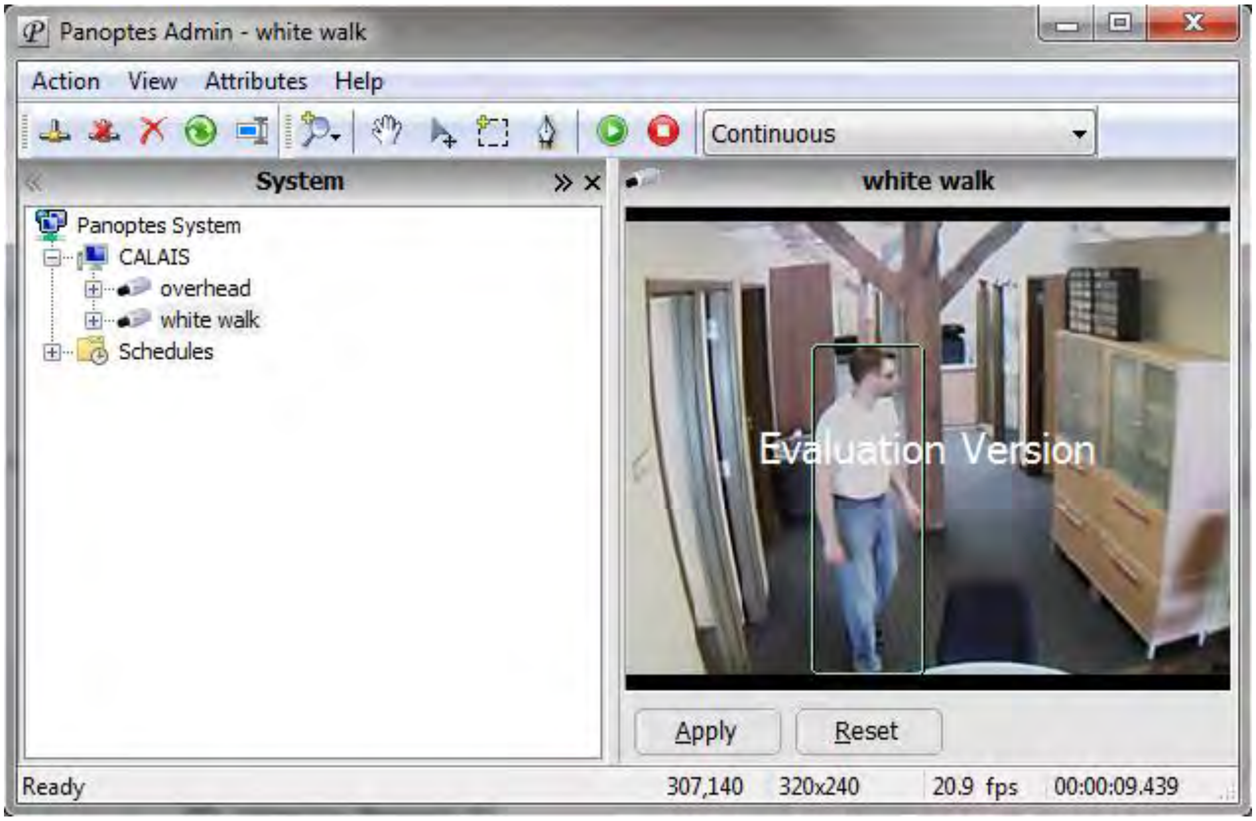


Figure 2: Screenshot of Panoptes

2.2 In-House Tracking System

As a form of risk mitigation, Primordial developed its own methods of analyzing raw video data and performing computer vision techniques on that data. Primordial purchased an inexpensive pan, tilt, zoom camera, the Foscam FI8918W as depicted in Figure 3, to use in experimentation. This camera has 300 degrees of rotation along the panning plane and 120 degrees along the tilt axis and is capable of outputting images in 320 X 240 pixels and 640 x 480 pixels.

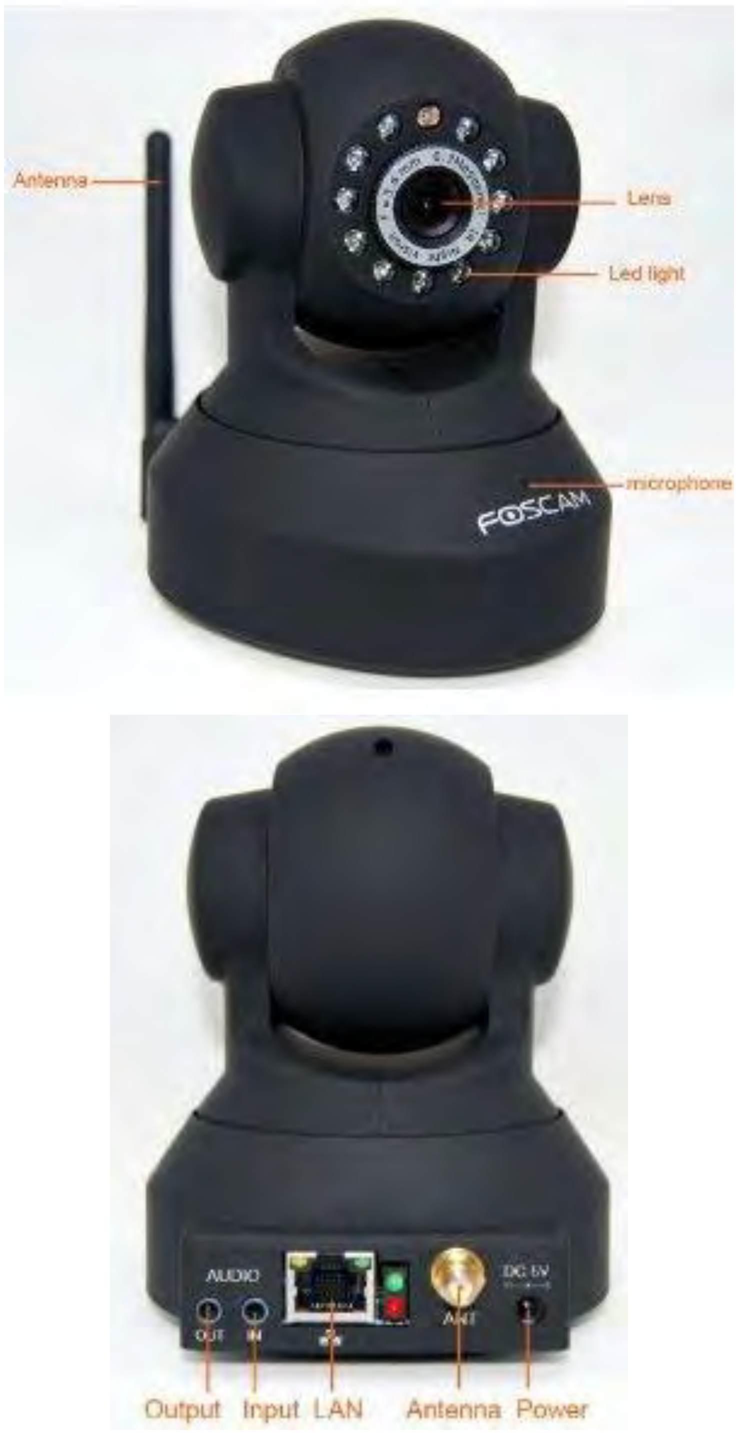


Figure 3: Foscam FI8918W Wireless IP Camera

The output of the camera's video frames is Motion JPEG. To facilitate rapid prototyping, we used EmguCV which is an open source C# wrapper for OpenCV.

2.2.1 Object Detection

In order to detect objects within the camera's field of view, we chose to start with a simple background subtraction algorithm. Since the RGB color space can be heavily distorted by changes in illumination, we operated on the HSV color space. The advantage of using the HSV color space is that shadows cast by objects detected do not cause many false positives. The algorithm was as follows:

(1.) Model the background by finding the pixel means for a sample set of frames
(2.) Get the current frame from the camera
(3.) Smooth out the current frame by using a Gaussian kernel
(4.) Get the absolute difference of pixels from the background model and the current frame for both the saturation and value channels of each pixel (see Figure 4).
(5.) Apply thresholding and morphology (dilation and erosion) to clean up noise and strengthen the area of interest in the foreground mask.
(6.) Find the intersection of the saturation and value foreground masks to further reduce noise and intensify the area of interest.
(7.) Find the contours of the foreground mask to identify the object of interest within the video feed (see Figure 5).

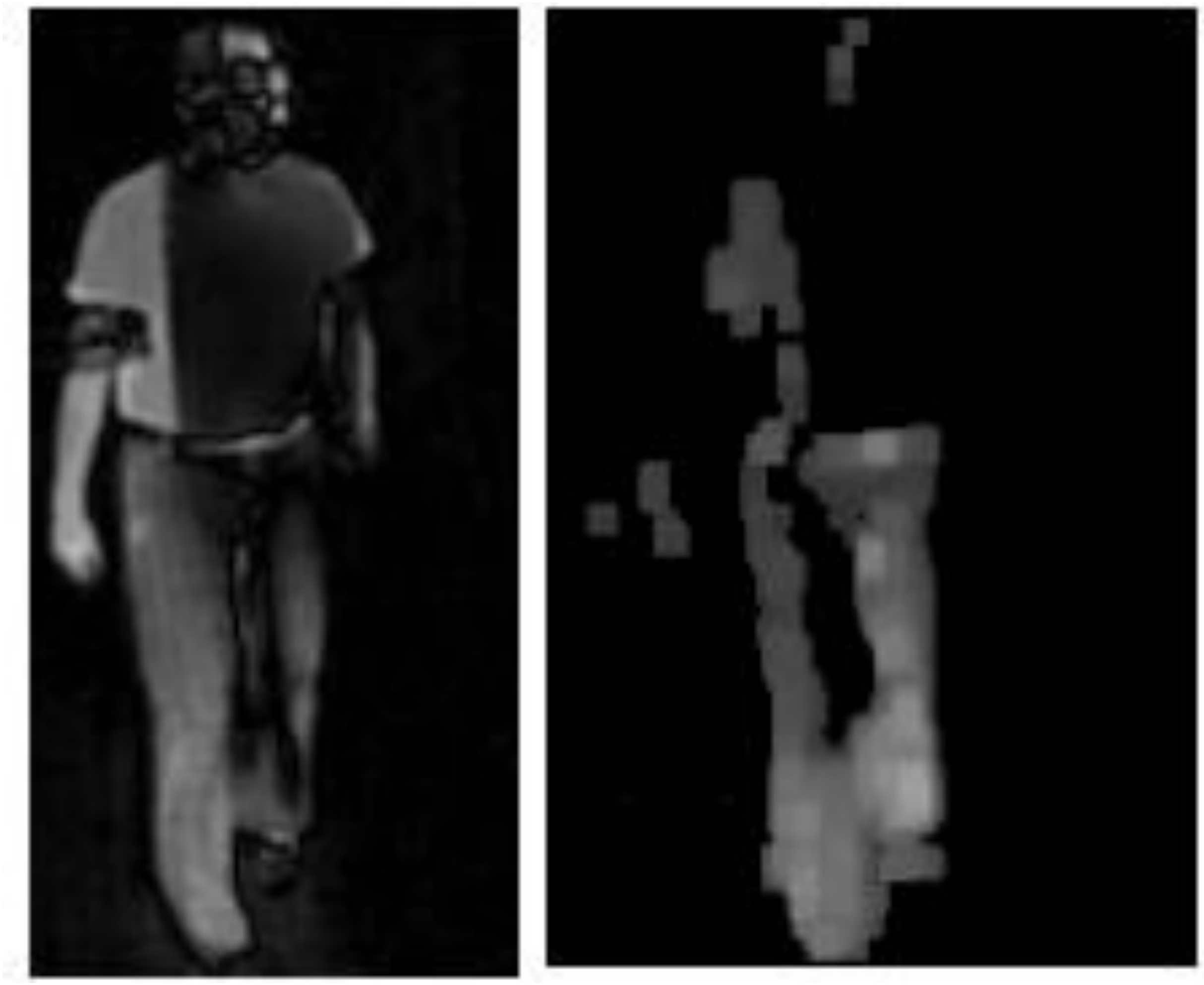


Figure 4: Absolute Difference of Pixels for Value and Saturation Channels

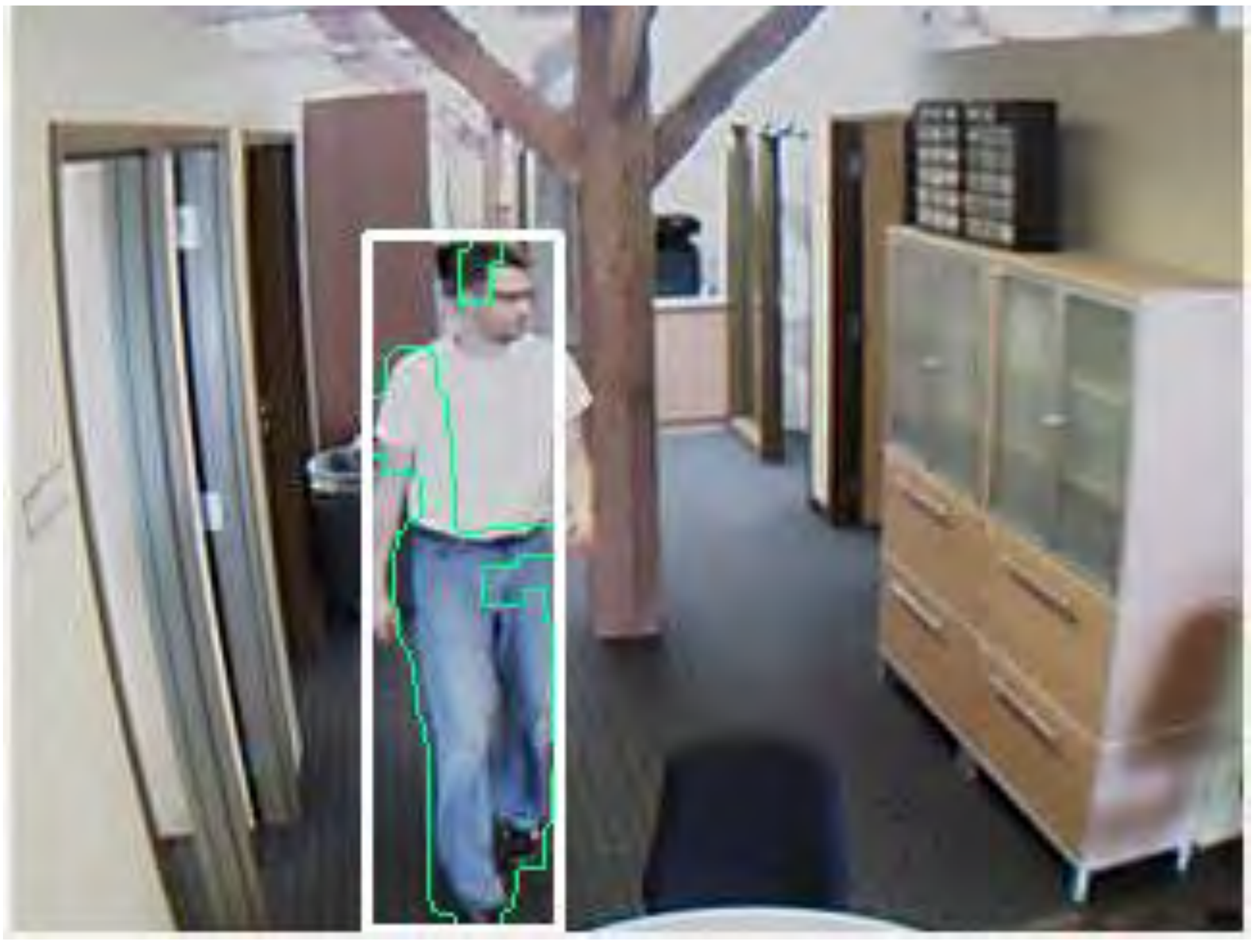


Figure 5: Final Image with Bounding Box Representing Area of Interest

2.2.2 Reduced Bandwidth

In an effort to reduce bandwidth of the video stream, we took advantage of OpenCV's ability to detect edges in a scene by using the Canny() method available. The Canny method reduces a color image to a gray scale image and then detects edges within the image by performing pixel analysis within a 3 by 3 square matrix. Values that are above the specified threshold are kept and then contours are formed from nearby neighbors. The image is cleaned up by setting the pixel values to the maximum allowed value, resulting in a black and white image exemplified by Figure 6 where white represents the edges detected.

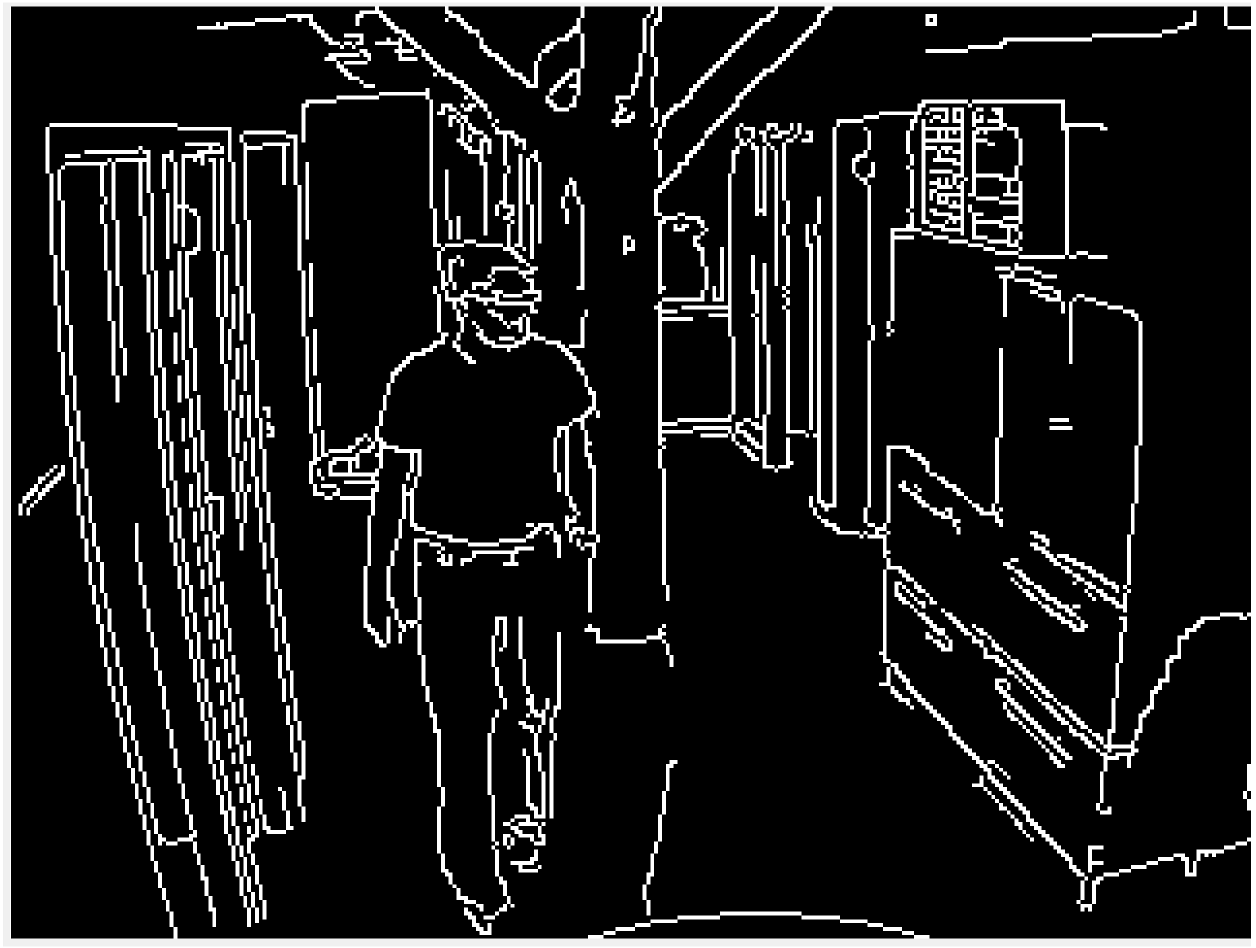


Figure 6: Canny Edge Image of Figure 5

2.3 Camera Calibration/Projection

To integrate into a user interface, Primordial needed to learn the camera's intrinsic and extrinsic calibrations, and the conversion from a 2D pixel location to a 3D real world location.

2.3.1 Intrinsic

To acquire the intrinsic parameters for the test camera, Primordial took 15 images of a chessboard. These images were fed into OpenCV's FindChessboardCorners() method. The method returned a set of screen points and real-world object points calculated from the chessboard images. The resulting sets of points were then used as parameters to the OpenCV CalibrateCamera2 method. This method created two files, one containing the intrinsic set of parameters and the other a set of distortion parameters for reversing image distortion.

2.3.2 Extrinsic

Extrinsic parameters for the camera were determined by providing a set of screen points and their corresponding 3D locations (in relation to a given reference point). These points, combined with the previously identified intrinsic parameters were used when calling OpenCV's FindExtrinsicCameraParams2, which produced a file containing the camera's extrinsic parameters.

2.3.3 2D To 3D Projection

OpenCV doesn't readily provide a method for converting from 2D to 3D coordinates, so we needed to create a system of linear equations to do this conversion. The basic issue with going from 2D to 3D coordinates is that for each 2D pixel location, there are infinitely many 3D locations which could correspond to the camera pixel. To solve for the actual 3D point, we need to add an additional constraint and assume that the 3D point is on the ground plane. So by combining the linear equation for the ground plane with the linear equations which define the relationship between 3D locations and camera pixels (i.e. based on the calibration matrices), we can solve the resulting system of linear equations for the desired 3D location.

Equation 1 presents OpenCV's implementation of translating 3D locations into 2D space, and it is the starting point for what we are trying to accomplish – going from 2D to 3D.

$$s\ m' = A[R|t]M'$$

or

$$s\begin{bmatrix}u\\v\\1\end{bmatrix} = \begin{bmatrix}f_x & 0 & c_x\\0 & f_y & c_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix} \quad \textbf{(1)}$$

Where (X, Y, Z) are the coordinates of a 3D point in the world coordinate space, (u, v) are the coordinates of the projection point in pixels. A is called a camera matrix or a matrix of intrinsic parameters. (cx, cy) is a principal point (that is usually at the image center), and fx, fy are the focal lengths expressed in pixel-related units. Thus, if an image from camera is scaled by some factor, all of these parameters should be scaled (multiplied/divided, respectively) by the same factor. The matrix of intrinsic parameters does not depend on the scene viewed and, once estimated, can be re-used (as long as the focal length is fixed (in case of zoom lens)). The joint rotation-translation matrix $[R|t]$ is called a matrix of extrinsic parameters. It is used to describe the camera motion around a static scene, or vice versa, rigid motion of an object in front of still camera. That is, $[R|t]$ translates coordinates of a point (X, Y, Z) to some coordinate system, fixed with respect to the camera.

Here are the detailed steps into how we translate 2D pixel locations into 3D locations.
Using the camera calibration matrix and distortion parameters, we use OpenCV to undistort each frame. Working with undistorted images allows us to use a pinhole camera model for the rest of the analysis and ignore the nonlinear distortion parameters.

To solve the equation for the ground plane, we need to define three known 3D locations on the ground. The form of the plane equation presented by Equation 2:

$$AX + BY + CZ + D = 0 \qquad \textbf{(2)}$$

We then solve for A, B, C, and D using OpenCV's cvSolve() function.

We then reorganize the original equation from Equation 1 to relate the 3D points to 2D camera location, resulting in Equation 3:

$$(u\ s,\ v\ s,\ s) = (\text{Intrinsic Matrix})*(\text{Extrinsic Matrix})*(X, Y, Z, 1) \qquad \textbf{(3)}$$

Where (u, v) is the known pixel point and (X, Y, Z) is the unknown 3D location.

We then combine the linear plane equation from Equation 2 with Equation 3 and use OpenCV's cvSolve() function to solve the system of linear equations, which results in our 3D point (X, Y, Z).

2.4 Integration into WorldWind

As an initial exploration into using NASA WorldWind and Pursuer, Primordial wrote a Java viewer for displaying target locations. The client software was a standalone program, but was intended for translation into a Java module for direct Pursuer integration. The application used the WorldWind SDK, a suite of tools for geospatial visualization, similar to Google Earth. The application allowed for presentation of icons and models to represent locations and full pan/tilt/zoom interaction with the displayed globe or map as well as extensive customizable overlays.

Initially, we used Collada models to represent target locations in order to differentiate between target types, but due to a known issue with WorldWind's Collada loader module, many models would not display or were missing parts or textures. We decided to use a generic WorldWind blob icon to represent target locations until those problems could be resolved.

The application used a simple server to receive target location information via a TCP connection. Target locations were sent from other Tentacle target tracking applications as simple latitude and longitude pairs delimited by commas. These were then parsed and a new target icon was created and placed in the WorldWind view screen, while any previous target location was removed, resulting in a smooth transition in the display. At the time, there was only functionality for tracking a single target for testing purposes.

2.5 Integration into Pursuer

Primordial requested and received a copy of the Pursuer software to begin integration efforts. However, due to complexity and lack of documentation for Pursuer, we decided to delay development on a module and focus on a standalone application.

2.6 Integration into Google Earth

As risk mitigation to the Pursuer integration, Primordial created a baseline Google Earth plugin to serve as an end-user interface if necessary. We then created a model of the Primordial office using Google Sketch Up. A small JavaScript application was created to allow us to insert an avatar into the screen and control the movement of that avatar by feeding geo-referenced points to the plugin. A looping demonstration of this can be seen at http://groundguidance.com/tentacle/. A still screenshot of the demonstration is given by Figure 7.

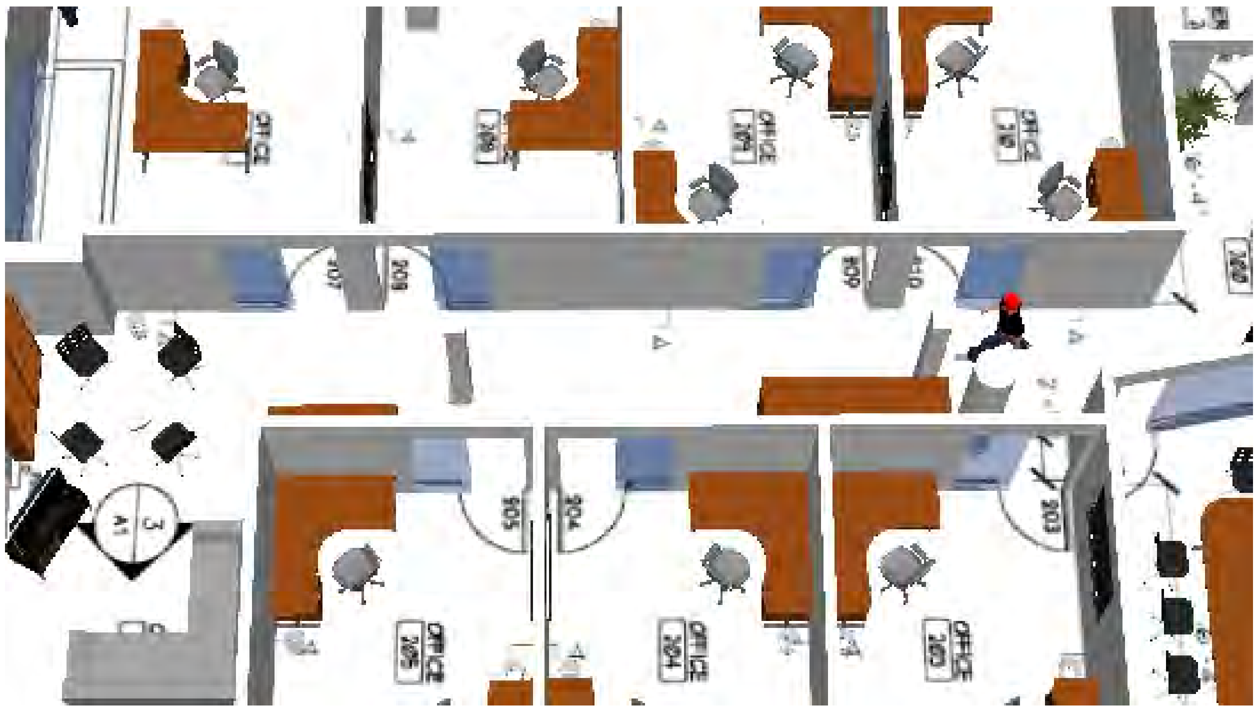
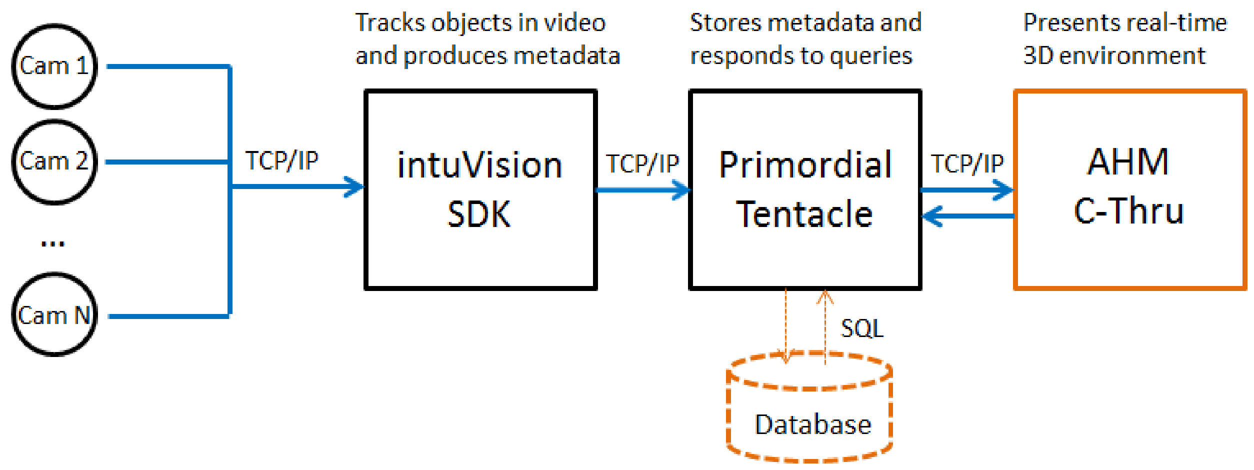


Figure 7: Google Earth Plugin for Tentacle

2.7 Initial Tentacle System Architecture

We worked on developing a complete architecture for the final Tentacle system. The intent was to facilitate performing queries on detected entities on both real-time data feeds and archived data. Figure 8 shows the proposed final architecture.



Tracks objects in video and produces metadata
Stores metadata and responds to queries
Presents real-time 3D environment
Cam 1
Cam 2
...
Cam N
TCP/IP
intuVision SDK
TCP/IP
Primordial Tentacle
TCP/IP
AHM C-Thru
SQL
Database

Figure 8: Tentacle System Diagram

2.8 Tentacle System Requirements Verification

We carefully extracted the project's requirements from the phase I solicitation, proposal, and various notes from phone calls and meetings. We then categorized each requirement into one of three major requirements areas: algorithms, database, and interface. Below, we outline the requirements falling into each of these major areas, and for each provide a description and means of verifying whether we met or will meet and how.

2.8.1 Algorithms

These requirements are for Tentacle's low-level algorithms for analyzing sensor streams, recognizing features, and producing streams of standardized, tagged metadata.

(1.) **Requirement 1: Take as input a multitude of raw data streams from varieties of sensors, including but not limited to video sensor streams.** In phase I, Tentacle takes as input video sensor streams only. In phase II, we will explore incorporation of additional sensor streams. For example, we will treat friendly C2 systems (e.g., FBCB2 and MTS) as sensor streams consisting of friendly-force positions. We will incorporate threat and event streams from the TIGR system. We will support dense streams (e.g., video) and sparse streams (geo-tagged pictures from digital cameras, akin to a single frame from a video). We will support other modalities (e.g., explicit audio recordings or audio tracks from existing video feeds). We will also explore performing text content analysis sources of written-word (e.g., RSS feed of local newspaper).

(2.) **Requirement 2: Produce as output a stream of tagged metadata representing recognized and extracted features.** Tentacle metadata streams all contain WWW information: what, where, and when. To answer "what", Tentacle includes two metadata tags: FeatureType and FeatureColor. Valid values for FeatureType are "Person" and "Vehicle", and valid values for FeatureColor are integer triplets in the RGB color space. Together, these metadata tags allow Tentacle to describe a feature like a "White Truck" or a "Blue Person". To answer "where", Tentacle will include metadata tags for the sensor's position (latitude, longitude, and height above ground level), orientation, and speed. To answer "when", Tentacle will produce timestamp metadata tags.

(3.) **Requirement 3: Associate metadata streams with their parent sensors and the supporting raw data from those sensors (supports future queries).** Tentacle metadata streams all contain information that describes the parent sensor, including make and model (e.g., Foscam FI8918W or AeroVironment Raven UAV), type (e.g., electro-optical or infrared video camera), resolution (e.g., VGA), and frame rate (e.g., 30 FPS). In addition, the metadata has pointers/identifiers for accessing the underlying raw sensor stream data.

(4.) **Requirement 4: Operate in near real-time (minutes, not hours or days).** Tentacle algorithms and components all operate in real-time for video streams of 30 FPS at VGA resolution.

(5.) **Requirement 5: Support operation using both live sensor streams and forensic (archived) sensor streams.** Tentacle currently supports interaction with live video feeds; in phase II, Tentacle will also support archived video feeds, as will be necessary for conducting queries.

(6.) **Requirement 6: Ensure output metadata is consistent in format and conforms to established schemas and standards.** We use the Cursor on Target XML schema.

(7.) **Requirement 7: Support multiple moving cameras.** In phase I, Tentacle supports stationary cameras. In phase II, Tentacle will support moving cameras mounted on UAVs. Support for moving UAV cameras will be accomplished by identifying appropriate object identification and tracking algorithms during phase I, and adopting/implementing those algorithms in phase II. We will leverage our unique business relationships with Carnegie Mellon University and AeroVironment, with whom we are teamed on a United States Army SBIR that deals with performing autonomous tracking of adversarial ground targets using the sensor payloads of teams of UAVs and UGVs.

2.8.2 Database

These requirements are for Tentacle's behind-the-scenes software framework and engine for storing metadata, querying metadata and producing results, and notifying of events.

(1.) **Requirement 1: Take as input streams of raw data and associated standardized tagged metadata.** Tentacle accepts raw video streams as well as metadata in the Cursor on Target format.

(2.) **Requirement 2: Persist metadata streams and raw data to a format suitable for long-term storage and query.** In Phase II, Tentacle will persist raw AVI video streams to disk and will persist metadata to a database accessible via SQL.

(3.) **Requirement 3: Accept as input queries on metadata and produce as output results that support user-level actions.** Tentacle currently accepts queries on FeatureType, FeatureColor for live entities; in Phase II, we will include Position, and Time as well as queries via SQL commands to the database.

(4.) **Requirement 4: Accept as input user-defined rules and automatically produce event notifications when rules are broken/triggered.** In Phase II, Tentacle will support rules via SQL commands to the database. Currently the intuVision alert system is employed, which stores alerts as XML configuration files that represent each camera's settings and rules.

(5.) **Requirement 5: Ensure high numbers of input streams can be accepted (high-scalability).** Tentacle can utilize low-bandwidth representations of sensor streams when possible to facilitate high sensor stream counts (24 cameras).

(6.) **Requirement 6: Adopt and support an open architecture for existing and future metadata standards.** We use the Cursor on Target XML schema.

(7.) **Requirement 7: Take as input streams of raw data and associated standardized tagged metadata.** Tentacle accepts raw video streams as well as metadata in the Cursor on Target format.

2.8.3 Interface

These requirements are for Tentacle's user-level GUI application for viewing sensor streams and metadata, creating queries, and receiving event notifications.

(1.) **Requirement 1: Take as input streams of raw data and associated standardized tagged metadata.** Tentacle accepts raw video streams as well as metadata in a Cursor on Target format.

(2.) **Requirement 2: Perform real-time representation (display) of metadata and indicate important features.** Tentacle presents a unified 3D environment using the NASA WorldWind rendering engine. The environment has terrain and man-made structures present, and is populated with avatars or icons that represent metadata entities. In Phase II, when employing avatars, we will use generic-looking 3D objects (e.g., a white pickup truck). When employing icons, we will use MIL-STD-2525B symbology.

(3.) **Requirement 3: Present a user-friendly interface for creating metadata queries against the framework.** Tentacle presents a collection of user interface widgets for constructing queries, including listing which metadata fields are available for query and what their acceptable parameters and ranges are. Tentacle's user interface minimizes the user's cognitive load.

(4.) **Requirement 4: Present a user-friendly interface for displaying results of queries and supporting likely user activities that follow.** Tentacle presents a collection of user interface widgets for enumerating and displaying results of queries and drilling down on a result for more details to conduct a useful task or action. Tentacle displays high-level results of queries (e.g., text results and video clip thumbnails, similar to YouTube) and allows the operator to drill down all the way to raw sensor streams by clicking on thumbnails.

(5.) **Requirement 5: Present a user-friendly interface for creating rules.** In Phase II, Tentacle will present a collection of user interface widgets for constructing rules and defining their parameters in terms of metadata tags.
(6.) **Requirement 6: Present a user-friendly interface for automatic notification of rule breakages and supporting likely user activities that follow.** Tentacle presents a collection of user interface widgets for displaying notifications and drilling down on a notification for more details to conduct a useful task or action. Tentacle's notifications are informative yet unobtrusive, and allow the operator to take useful action (e.g., obtaining coordinates or verifying an incident).
(7.) **Requirement 7: Support an outdoor UAV-based demonstration scenario.** In Phase II, we will seek to conduct a live outdoor demonstration on actual UAV hardware. To accomplish this, we will leverage our unique business relationships with AeroVironment and Carnegie Mellon University to obtain UAV hardware and software.

2.9 Internal Demonstration

On 27 May 11, we conducted the first live demonstration of the Tentacle system for an internal audience. The demonstration was simple, but designed to be a building block for future demonstrations. Specifically, the first internal live demonstration had the following characteristics:

- **Single-camera.** Only one camera (a Foscam FI8918W) was taken as input. This camera uses MJPEG compression and outputs VGA frames at 15 FPS or QVGA frames at 30 FPS. For our purposes, we use the QVGA setting. Utilizing one camera for the first internal live demonstration, instead of many, allowed us to keep the scenario simple and thus achieve an earlier demo. Later, we would integrate a second camera into our Tentacle system; section 2.3 has details.
- **Single-entity.** As exemplified by Figure 9, only one entity was tracked at a time. Recognizing entities by assigning each a unique ID in a programmatic fashion is a difficult image tracking problem, and therefore it did not make sense for us to implement our own algorithms for doing this, as we planned to rely on our subcontractor, intuVision, for this capability.
- **Indoor.** The demonstration took place inside of Primordial's offices. We planned to conduct later demonstrations in outdoor environments; however, the indoor environment made it easier to hook up our cameras and computers to power and Ethernet.
- **Google Earth.** Entities were rendered as 3D objects in Google Earth. While we explored integration with NASA WorldWind previously, we chose to use Google Earth for development due to our past experience developing with it, and the maturity of the Tentacle user interface mockup we created (located at http://groundguidance.com/tentacle).
- **No queries yet.** The user was unable to query live or archived footage. We plan to design and develop user interfaces and algorithms for allowing users to perform real-time queries on both live and archived video footage.

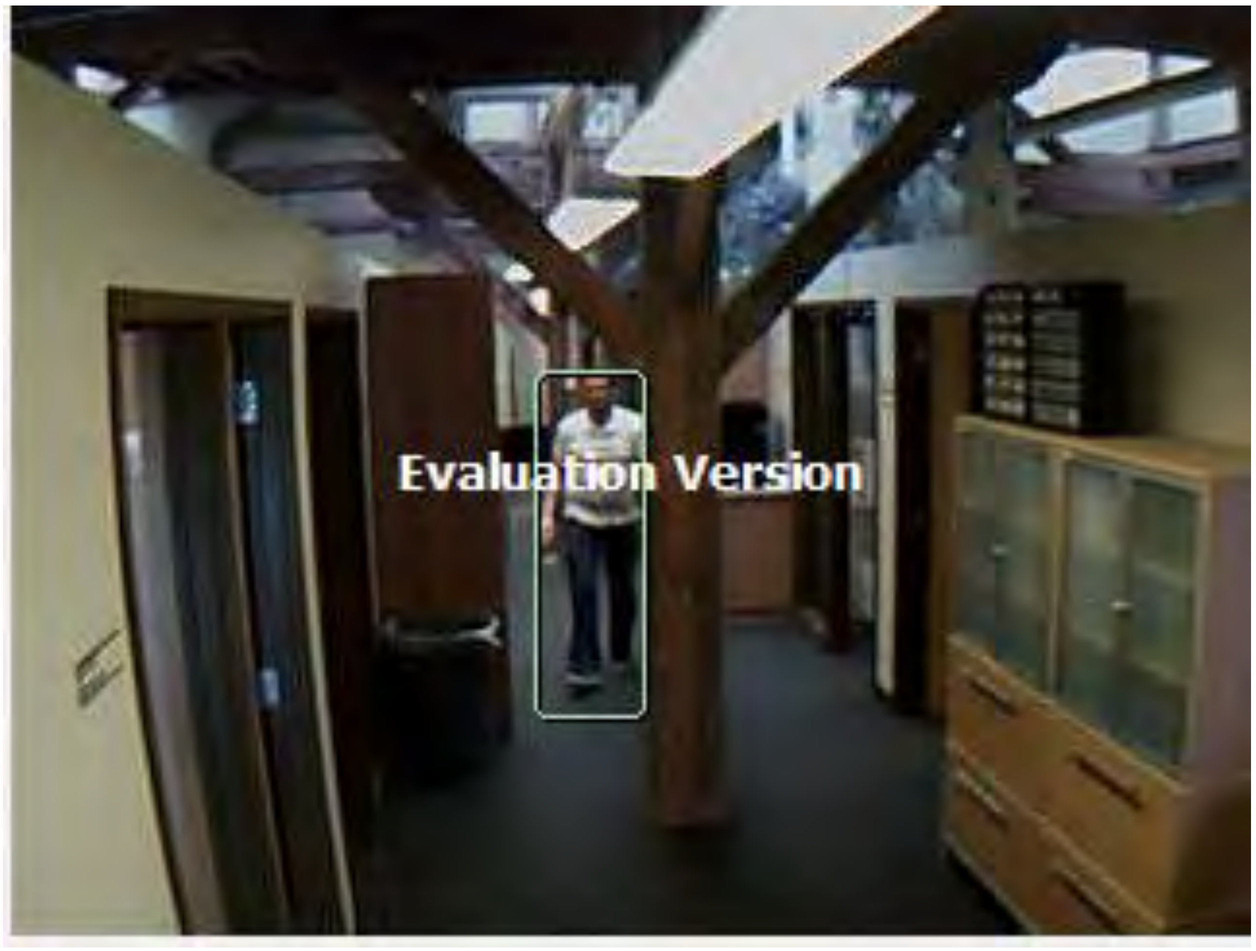


Figure 9: Single-Entity Tracking, and View from First Camera

We utilized our own object tracking algorithms for the first internal live demonstration. These algorithms perform object tracking by establishing a background model and using background subtraction on subsequent frames to extract the foreground. This works reasonably well in environments where the background changes slowly or infrequently, such as the inside of our offices. The reason we used our own object tracking algorithms, as opposed to using those of our partner, intuVision, was that intuVision's software development effort for Tentacle was still underway (we would later receive their intuVision SDK and incorporate it into Tentacle).

During this demonstration, Tentacle analyzed the incoming video feed to locate motion, and then determined the intersection of the detected entity with the ground plane. The object tracking algorithm implemented background subtraction to extract the mask of pixels that indicate motion. Once the mask was created, we performed image processing to clean up noise and strengthen the region of motion. We then created a bounding box around the region of motion which allowed us to locate the center of mass of the entity. From this we could determine the horizontal pixel coordinate and the bottom-most vertical pixel coordinate of the entity to determine the intersection of the entity with the ground plane. The detection system then sends those pixel coordinates to the Tentacle user interface to be transformed into geographic coordinates, at which a 3D entity avatar is positioned. Tentacle listens for entity data packets over a TCP/IP connection, where each data packet contained the horizontal and vertical pixel coordinates of the detected entity. Tentacle then used the camera's intrinsic and extrinsic parameters to solve the linear equation (outlined previously) to transform the pixel coordinates to geographic coordinates. Tentacle then invoked JavaScript to update the Google Earth widget move the avatar to the appropriate location.

2.10 Integrated intuVision SDK into Tentacle

On 23 May 11, intuVision delivered their SDK. intuVision's SDK powers the intuVision Panoptes software, which we evaluated and discussed in Interim Report #1. Upon receiving the SDK, we began integrating its functionality into Tentacle. Primarily this consisted of swapping out our object detection and tracking algorithms with calls to the intuVision SDK application programming interface (API).

First, Tentacle instantiates one instance of the intuVision SDK for each camera. In our current system, Tentacle accepts input from one or two cameras which both feed their data to a single computer running two instances of the intuVision SDK (one for each camera). Initialization of the intuVision SDK is done through the "InitializeStandAlone", "SetCameraParameters", and "SetCallback" functions, whose method signatures and documentation are provided below:

```
/// <summary>
/// Initializes a standalone Panoptes SDK instance
/// </summary>
/// <param name="xmlPath">XML settings associated with the camera</param>
/// <param name="licenseKey">intuVision License Key for this installation</param>
/// <param name="logFile">Path to log file, or null for no log</param>
/// <param name="state">The initial state of the camera</param>
/// <returns>A pointer to a valid intuVision Handle</returns>
public static void* InitializeStandAlone(string xmlPath, string licenseKey, string logFile,
intuVisionCameraState state)

/// <summary>
/// Sets/Updates the cameras parameters for an instance of Panoptes
/// </summary>
/// <param name="handle">The handle to the instance of Panoptes</param>
/// <param name="xml">The updated camera settings in xml formatted string</param>
/// <returns>1 upon success</returns>
public static int SetCameraParameters(void* handle, string xml)

/// <summary>
/// Sets a callback function for the intuVision SDK to communicate
/// </summary>
/// <param name="handle">The handle for the intuVision instance</param>
/// <param name="function">The callback function to be executed</param>
/// <returns>1 upon success</returns>
```

```
public static int SetCallback(void* handle, Delegate function)
```

Tentacle receives the raw video stream from each video camera. However, after Tentacle receives each frame, it sends it to the intuVision SDK for processing, by calling the "SetNewFrame" function, whose method signature and documentation is provided below. Note that the data format for the frame being passed to the intuVision SDK is simply a pointer to an image file (representing the frame) residing in local memory.

```
/// <summary>
/// Passes a new frame to be handled by the intuVision SDK
/// </summary>
/// <param name="handle">A pointer to the instance of the intuVision SDK</param>
/// <param name="pData">A pointer to the image data</param>
/// <param name="width">The width of the image</param>
/// <param name="height">The height of the image</param>
/// <param name="seconds">The number of seconds elapsed since Epoch</param>
/// <param name="milliseconds">The number of milliseconds</param>
/// <returns>1 upon success</returns>
public static int SetNewFrame(void* handle, byte* pData, uint width, uint height, UInt64
seconds, ushort milliseconds)
```

Thus, the intuVision SDK takes video frames (which are nothing more than still images with a resolution equal to that of the camera) on a frame-by-frame basis. Internally, the intuVision SDK implements a series of algorithms that allow it to reliably detect and track motion from frame to frame. The detected motion is represented by an "intuVisionObject" object. In addition, with each passing frame, the intuVision SDK also evaluates a series of rules to determine if any alerts should be generated (e.g., a person stepped into a particular region of interest). Each such alert is represented by an "intuVisionAlert" object. These intuVisionObject and intuVisionAlert objects are packed into a containing "intuVisionFrame" object, which is transmitted from the intuVision SDK to Tentacle for each frame that Tentacle sends. The properties of the intuVisionFrame, intuVisionObject, and intuVisionAlert objects are presented in Table 1, Table 2, and Table 3 respectively.

Table 1. Properties of the intuVisionFrame Object

Property	**Description**
Data	Pointer to the image data (BGRA format)
Width	Width of frame image
Height	Height of frame image
TimeStamp	Timestamp of frame in seconds since Unix epoch
Milliseconds	Millisecond portion of timestamp
Alerts	Pointer to array of current alerts
numAlerts	Number of alerts in array
Objects	Pointer to array of current objects
numObjects	Number of objects in array

Table 2. Properties of the intuVisionObject Object

Property	**Description**
LocationX	Bounding box X location (Upper Right)
LocationY	Bounding box Y location (Upper Right)
Width	Bounding box width
Height	Bounding box height
UID	Unique ID for object
Type	Object type (Person, Vehicle?)
Confidence	Confidence level of the classification
Property	Description

Table 3. Properties of the intuVisionAlert Object

Property	**Description**
timeStart	The start time of the timestamp in ticks
elapsedTime	The elapsed time of the alert in milliseconds
Name	The name of the alert
Type	The alert type
Description	Description of the alert
Priority	Alert priority
UID	The alert’s unique ID
Confidence	The confidence level of the alert 1-100
ObjectUIDs	Pointer to an array of ObjectIDs associated with this alert

2.10.1 Determined Computational Platform Requirements

During the teleconference for Interim Report #1, we recorded an action item to determine the computational platform requirements for running Tentacle, which will largely depend on the requirements for the intuVision software. These minimal requirements depend on video resolution, format, and the business of the content of the scene, each of which factor into how much processing power is consumed by decoding the video. In general, a mid-range desktop (dual-core, 2.13 GHz, 2 GB RAM) can run six, seven, or eight low-resolution (640 x 360) cameras. Lower-end mini-PCs (example provided in Figure 10), if they employ the Nvidia Ion architecture, can run two, three, or four cameras.

Figure 10: Example of a Mini-PC That Employs the Nvidia Ion Architecture

The most important tech spec is that the machine should include a CUDA-capable GPU, which allows the offloading of video processing onto the GPU, even if the GPU is only mid-range (such as what comes standard in a business desktop).

2.10.2 Tasked intuVision With SDK Upgrades to Support Advanced Features

Since funding was still available to intuVision, we requested that they upgrade their SDK with the following features:

(1.) **Ability to classify an object as a person or vehicle.** The reason for this feature was to allow objects to be represented in the 3D environment with the correct avatar. Furthermore, objects classified in this manner also have a confidence level associated with them. We can use this confidence level to threshold out false positives which may be caused by other changes in the environment.

(2.) **Ability to extract colors from an object.** This will provide us with the initial basic metadata to be able to perform queries on detected objects, e.g., persons with white shirt and black pants. Additionally, this will give us the ability to perform rudimentary identification of entities that temporarily leave a camera's FOV and perform camera handoffs of an entity.

2.11 Demonstrated Support for Multiple Cameras

On 17 Jun 11, we conducted our second live demonstration of Tentacle, again for an internal audience. This demonstration involved adding a second camera to the Tentacle system, the placement of which in our office space is shown in Figure 11 and the view from which is shown in Figure 12.

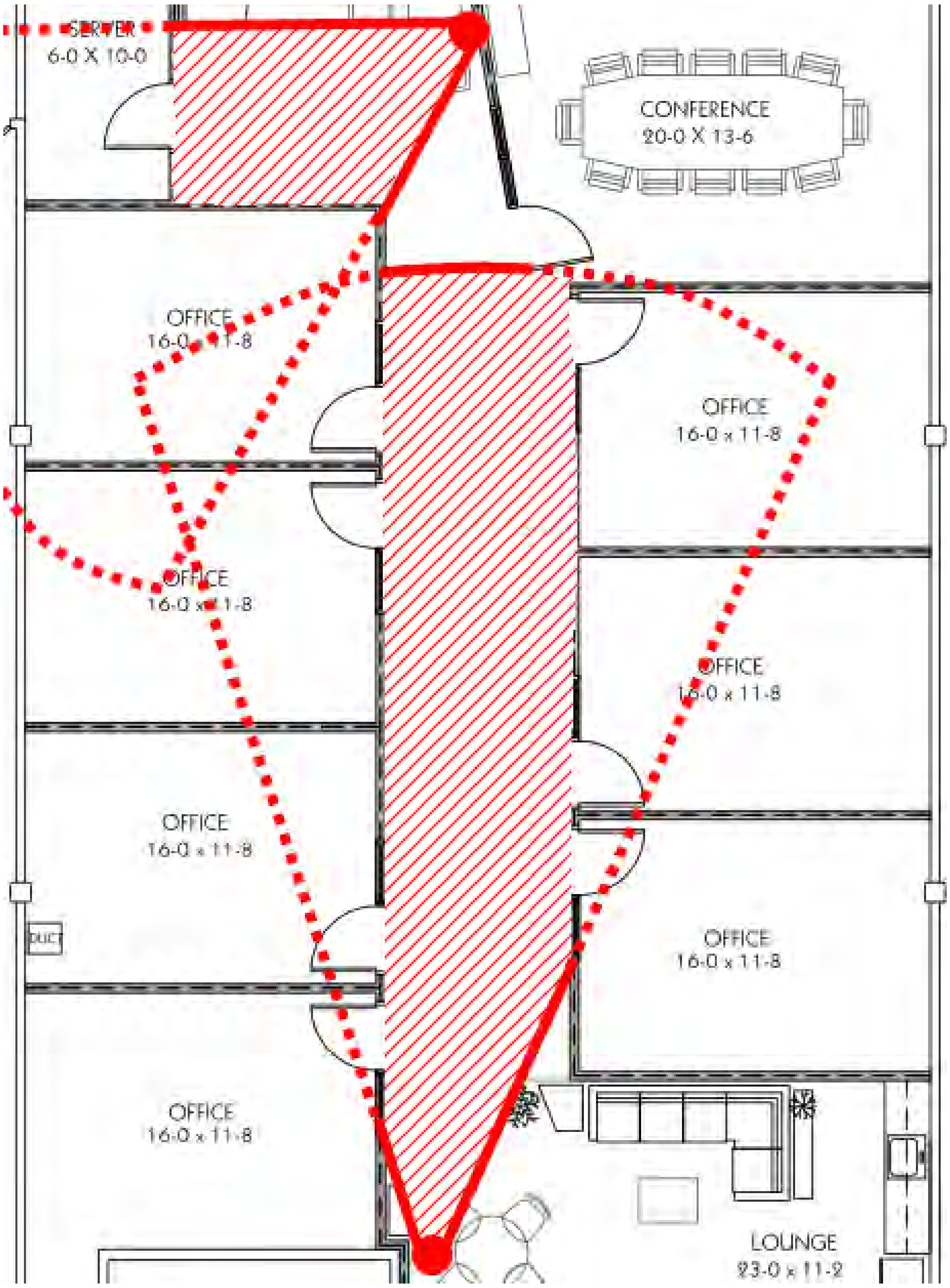


Figure 11: Approximate Location and FOV of Second Camera

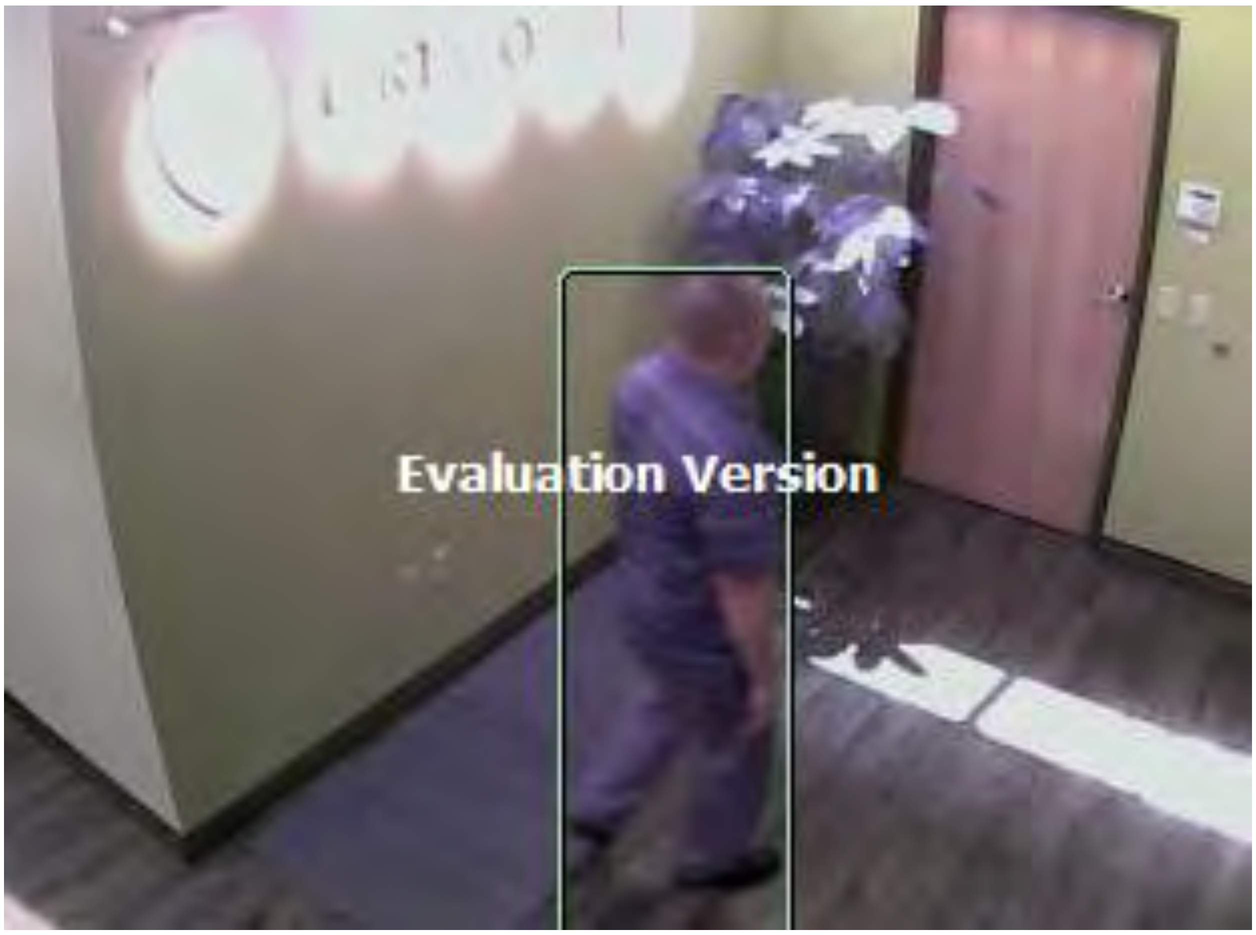


Figure 12: Actual View from Second Camera

At this point we chose to do some re-architecture of the Tentacle source code to increase flexibility in adding cameras/sensors to the system. Both the detection system and GUI were upgraded to accept multiple camera feeds. In order to support multiple cameras, each camera needs to have a unique identifier so that all parts of the system can determine where the information is coming from. On the detection system, the software maintains a hash table of camera objects connected to that machine. Each camera object on this machine is responsible for connecting itself to the GUI TCP/IP socket and transmitting data. On the GUI, a similar hash table contains all the camera objects connected to the entire system. The GUI camera object contains its intrinsic and extrinsic parameters, its geographic location, and the camera object name. It is important that the camera names match up from the detection system to the GUI. Additionally, separating the camera's by their name allows the user to request the video feed from the camera.

The TCP/IP packet information was then updated to include the camera name. When the GUI receives the packet, it determines what camera sent the information by looking up the camera based on the camera name in the hash table. The GUI then pulls the correct information for that particular camera to transform the horizontal/vertical pixel coordinates into geographic coordinates. Once the GUI invokes the JavaScript, the avatar is placed in the correct location.

2.12 Demonstrated Support for Simultaneous Tracking of Multiple Entities

On 30 Jun 11, we conducted our latest live internal demonstration. This iteration of Tentacle involves tracking multiple objects in all camera views, as depicted in Figure 13.

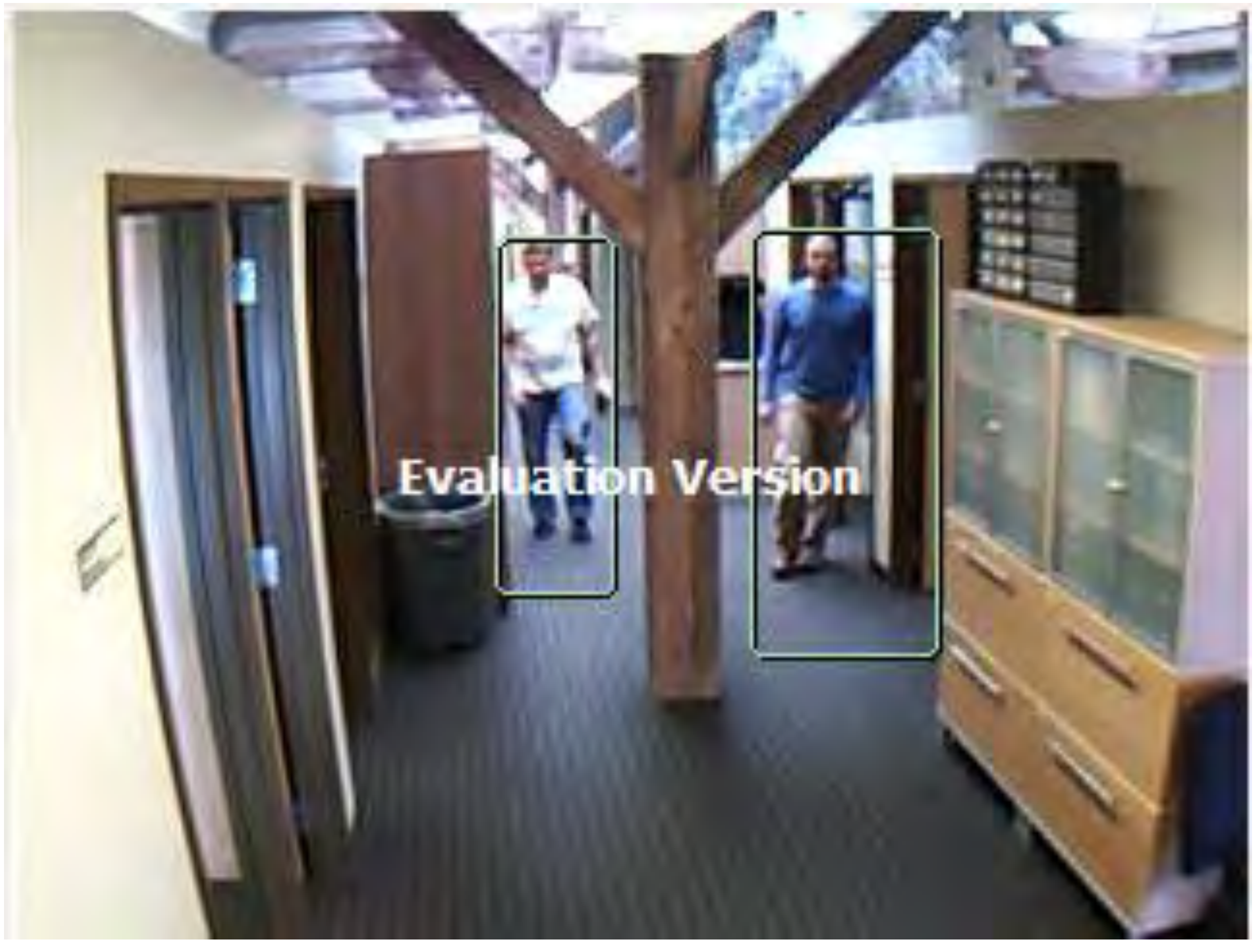


Figure 13: Multi-Entity Tracking

We upgraded the TCP/IP information packet to send the following information:

- The number of objects detected
- The number of events detected
- A list of Detected Objects
- A list of Detected Events
- The name of the Camera

By constructing our packets in this manner, we are able to lump all of the detected activity within a frame and send it once instead of sending many small packets. This allows us the flexibility to schedule the rate of packets flowing into the TCP/IP server of the GUI. Once the GUI receives this packet, it then deconstructs it into its individual components. For each object in the packet, the GUI checks to see if it has previously seen this entity. If it has, it will update a previously assigned avatar to move to the correct location. If not, it will create a new avatar for this object and place it in the appropriate spot. The GUI also updates a hash table with the current timestamp to track when each object was last seen in a camera field of view. By tracking this, we are able to remove stale objects from the 3D representation of the world and we are also able to attempt to re-identify previously seen entities that transfer between one camera's field of view and another's or an entity that repeatedly enters and exits one camera's FOV. If an entity hasn't been seen again for a significant amount of time, then the system will drop that entity from its hash tables.

2.13 Investigated Open System Architectures and Standards

One key requirement for the Tentacle system in phase II and beyond is the employment of an open system architecture and framework. As such, we spent time investigating what, if any, existing open standards Tentacle might employ to fulfill the requirement. We came across dozens of relevant and interesting documents, but will only discuss the most relevant one: STANAG 4559 AIR (EDITION 3) – NATO Standard ISR Library Interface. Our discovery of STANAG 4559 led us to perform a re-architecture of Tentacle, one that employs explicit content provider (sensor), server, and client nodes.

2.13.1 STANAG 4559

STANAG 4559, NATO Standard ISR Library Interface (NSIL Interface, or NSILI), is a standard interface for querying and accessing a library of ISR products, or IPL (ISR product library). The principle role of an IPL is to provide a search function that provides users visibility to all ISR library product information. The NSILI provides a standard "bridge" between the user interface that supports this search and display of library product information, and the IPL itself. Multiple users can simultaneously reference or execute identical queries. After reviewing the search results, selected library holdings may be requested for delivery.

2.13.1.1 User

In STANAG 4559, a user is defined as the person invoking the data request interactions with the library. For the Tentacle project, users may be Air Force intelligence analysts, imagery analysts, cartographers, mission planners, and simulations and operational users.

2.13.1.2 Client

In STANAG 4559, the client is the software application that allows the user to interact with the IPL by providing a mechanism for translating the user's requests into the correct form for transmission to the IPL and for translating the responses from the IPL into a suitable form for presentation to the user.

The user interface implemented by the client is outside the scope of STANAG 4559, but is certainly within the scope of the Tentacle project. For Tentacle, we have experimented with a user interface that utilizes a Google Earth plugin for presenting imagery and overlays.

2.13.1.3 ISR Product Library (IPL)

In STANAG 4559, the IPL is responsible for storing ISR products and responding to user requests. Users retrieve information from the IPL by submitting a user-generated query. To identify and locate library information, users perform searches via queries of metadata contained within library indices. Once the information is located, users will review metadata and preview the information before requesting product delivery.

2.13.1.4 Metadata

STANAG 4559 specifies metadata model entities and views that are capable of strong metadata for several relatively independent categories of ISR products. STANAG 4559's metadata model is comprised of several layers:

- Minimum Layer
- Reporting Layer
- Collection Coordination and Information Requirements Management (CCIRM) Layer
- Experimental Layer

The Minimum Layer supports ISR product types of Imagery, Video, and GMTI (Ground Moving Target Indicator). The Experimental Layer supports Tactical Data Link ISR products. As such, the Minimum Layer and Experimental Layer are the most relevant for Tentacle.

2.13.1.5 Queries

STANAG 4559 defines how the IPL can be queried by clients. At a high level, a query consists of a message sent from the client to the IPL, and a response message from the IPL to the client describing which, if any, of the IPL's holdings meet the query's parameters. A simple example would be a query for a video clip from 1997. If the IPL has one or more such video clips, it sends a list of those clips to the client for further processing.

At a lower level, a query begins with a query being constructed and transmitted to the IPL, where it is parsed. Both user and IPL will have an interest in the query being syntactically correct. The user checks are assumed to occur prior to transmitting the query and will use knowledge of the IPL parameters "discovered" during the login process. The syntax of the query is the Boolean Query Syntax (BQS) as defined in Annex F of STANAG 4559. A Results Profile, comprising a list of desired attributes to be returned including sorting preferences, is sent along with the query. The IPL checks are performed and the client will be notified if the query is invalid. A validated query will be allocated a reference that will be made known to the client. Given a syntactically valid query, the system performs the search for matching items in the library at its disposal. Upon completion of the search, the system produces query results, which are sent to the client according to the results profile. In the client, the query results are then organized and managed as required, according to the functionality provided by the client implementation. The user can then view the results and decide how to continue. The query process can be executed iteratively until the user has identified a set of products that they wish to order, at which time the order process is invoked.

2.13.1.6 Orders

STANAG 4559's order process occurs when a user has identified a product to be viewed from the IPL's holdings. The order process is similar in many ways to the query process but the order data structure is different from the query data structure. An order will specify a set of product identifiers that uniquely define the required products. These product identifiers are returned as part of the earlier query process. To request delivery of a product or set of products, an order is constructed by the user and submitted to the IPL where it is parsed. As with queries, both user and Library will have an interest in the order being syntactically correct. In addition to product identifiers, as part of the order the user specifies (for each product) the desired product format including data format and compression, a delivery destination and delivery method (e-mail or physical), details of any alterations to be made or tailoring to be done prior to delivery, delivery priority and, for physical deliveries, the required media type. Several aspects of creating the order could be automated by the client application, perhaps using a standard profile. The client will be notified if the order cannot be filled as requested or if any part of the order is invalid. Products may also be retrieved using a metadata field that provides a handle for direct access to the product, such as a URL for https transfer. Product retrieval via the direct access handle does not allow any request for product alteration, compression and so on.

2.13.1.7 Subscriptions

The subscription process is in STANAG 4559 outlines a way for a client to be automatically notified when new content (meeting user-defined criteria) is added to the IPL. The subscription process is similar to the query process because a client needs to send a query to the IPL, which will select the products that match the query. In addition, the IPL will continue to monitor itself for new products being added and add those products that match the query to the query results. The query creation is the same as described for the query process. The syntax of the query is the Boolean Query Syntax (BQS). The Results Profile provided by the client is handled in the same way as for the query process. In addition to the basic query parameters, the client provides a query lifespan. This defines when the monitoring of new products starts and ends, and at what interval rate are new matching products reported. The library can directly inform a client of new matching products by using a callback. Alternatively the client can poll the server.

2.14 Revised Tentacle Architecture to Employ a Client-Server Model

As of Interim Report #2, Tentacle employed a monolithic system architecture where each major functional component was run together on one computer. This included intuVision's SDK for processing video feeds, code for converting from 2D video pixel coordinates to 3D geographic world coordinates, and a GUI based on Google Earth for presenting entity avatars. This architecture, while it served well for demonstrating basic Tentacle concepts, has several disadvantages moving forward.

First, query support will be difficult without a well-defined, centralized database in which to archive video and associated metadata. Second, running all components on a single computer is unrealistic; in reality, sensors are remote and provide output to a command center, which may then re-transmit data to other locations. Third, high-cohesion and tight-coupling between Tentacle's software modules would likely cause unnecessary defects in the software in phase II, as changing one module (e.g., the 2D to 3D conversion code) may immediately break several other modules.

To address this, we devised a service-oriented system architecture called Tentacle System that is comprised of three components:

(1.) One or more Tentacle Mote (TM) nodes
(2.) A single Tentacle Server (TS) node
(3.) One or more Tentacle Client (TC) nodes

Each of these components is described and diagrammed in detail below, and are largely based on the components outlined by STANAG 4559.

2.14.1 Tentacle Mote

As diagrammed in Figure 14, a Tentacle Mote is comprised of a single video camera and a computer running the intuVision SDK. A Tentacle System may be comprised of one or many Tentacle Motes. In a Tentacle Mote, the camera sends motion imagery frames to the computer over a TCP/IP connection on a LAN. The computer then performs frame-to-frame analysis to identify entities and track their movement through the scene. The output of a Tentacle Mote is simply the output of the computer, which is streaming motion imagery with associated metadata describing entities in the scene.

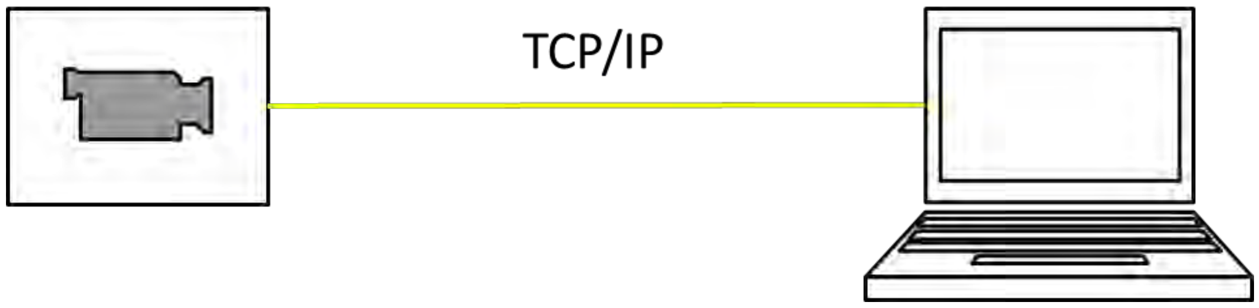


Figure 14: Tentacle Mote Comprises a Video Camera and a Computer

Tentacle Mote is an engineering black box, meaning that other Tentacle System components need not care about Tentacle Mote's inner workings, so long as the output produced by Tentacle Mote is as specified. That said, to keep costs low in phase I, we are using a video camera separate from a computer which performs image processing. An alternative would be to use a smart camera as exemplified in Figure 15, which combines the camera and computer into a single piece of machinery.

Figure 15: Sony Smart Camera Running Windows XP Embedded Priced at $4,754

In phase II and beyond, a Tentacle Mote might be comprised of an AeroVironment Raven UAV system, consisting of an air vehicle with an on-board video camera and a ground control station laptop that processes the UAV's sensor data.

2.14.2 Tentacle Server

As diagrammed in Figure 16, Tentacle Server represents the heart of the Tentacle System. A Tentacle System consists of a single, centralized Tentacle Server. A Tentacle Server consists of a computer workstation with sufficient hard drive space to archive video and metadata collected from Tentacle Motes. The output of Tentacle Server is provided to Tentacle Clients who connect to the Tentacle Server. The Tentacle Server has two major sub-modules, both facilitating independent interaction with Tentacle Client nodes. The first sub-module is the Tentacle ISR Product Library (TIPL), and the second is the Tentacle Tactical Data Link (T2DL). These sub-modules are described in detail as follows.

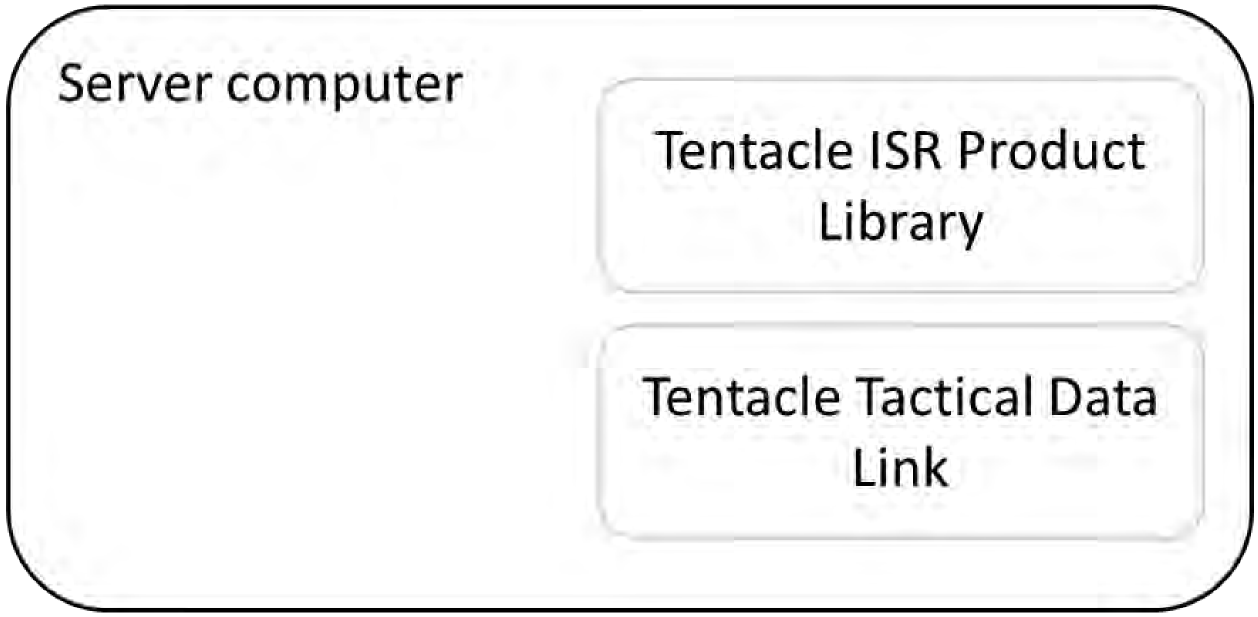


Figure 16: Tentacle Server Consists of a Server Computer with Two Software Sub-Modules

2.14.2.1 Tentacle ISR Product Library

As Tentacle Server collects ISR information (i.e., video and metadata) from Tentacle Motes, the TIPL places that ISR information into a library and catalogs it for future query and access, thus supporting archived operation. TIPL supports Tentacle System's automatic alert functionality by monitoring incoming ISR data, evaluating User-created conditions and rules, and sending an asynchronous notification to Tentacle Client when a positive match is found. In addition, the TIPL acts as a relay for streaming the incoming live video and metadata to Tentacle Clients, thus supporting live operation.

The concept for TIPL comes from STANAG 4559, which specifies a standard interface for an ISR product library. Thus, TIPL embodies both the library itself, implemented as database and video storage mechanisms on disk, and the interface to that library, represented as a web server listening for Tentacle Client connections on a certain port, over which standard API traffic will flow.

Tentacle Server accepts live video on one port and associated metadata on a separate port. Both sources of data originate from Tentacle Motes. The data streaming into the live video port is intuVision Frame objects in a binary format, and the data streaming into the metadata port is in the Cursor on Target format.

2.14.2.2 Tentacle Tactical Data Link

As Tentacle Server collects ISR information (video with associated metadata) from Tentacle Motes, T2DL processes the metadata into a tactical data link to be made available as a live stream to Tentacle Clients. The content of this live stream consists of messages with each message containing entity geographic position, type, color, and time stamp (note that this data stream functions independently of any video source). In this manner, T2DL allows Tentacle Clients to have the option to consume only a lightweight data feed representing battlefield entities. T2DL is modeled after existing tactical data links like Link 16.

2.14.3 Tentacle Client

As diagrammed in Figure 17, Tentacle Client represents the User's interface to entire Tentacle System. A Tentacle System may be comprised of one or many Tentacle Clients. A Tentacle Client consists of a computer workstation or laptop running an application. The Tentacle Client uses NASA WorldWind for visualization of the 3D environment.

In phase II, we will build a Tentacle Client using the Panoptic C-Thru user interface, which will provide major benefits with respect to 3D environment and avatar modeling.

Figure 17: Tentacle Client Consists of a Computer Workstation

2.14.3.1 Display of Real-Time T2DL Feed Data

Tentacle Client monitors incoming real-time T2DL traffic from Tentacle Server and displays 3D avatars for each entity on the WorldWind globe. In this sense, Tentacle Plugin is treating T2DL data no differently than other tactical data links such as Link 16, Blue Force Tracker (BFT) as depicted by Figure 18, or Ground Moving Target Indicator (GMTI). In the case of Tentacle, the entities are detected by video cameras; in the case of Link 16 and BFT, by active transponders placed on friendly vehicles; and in the case of GMTI, by synthetic aperture radar (SAR).

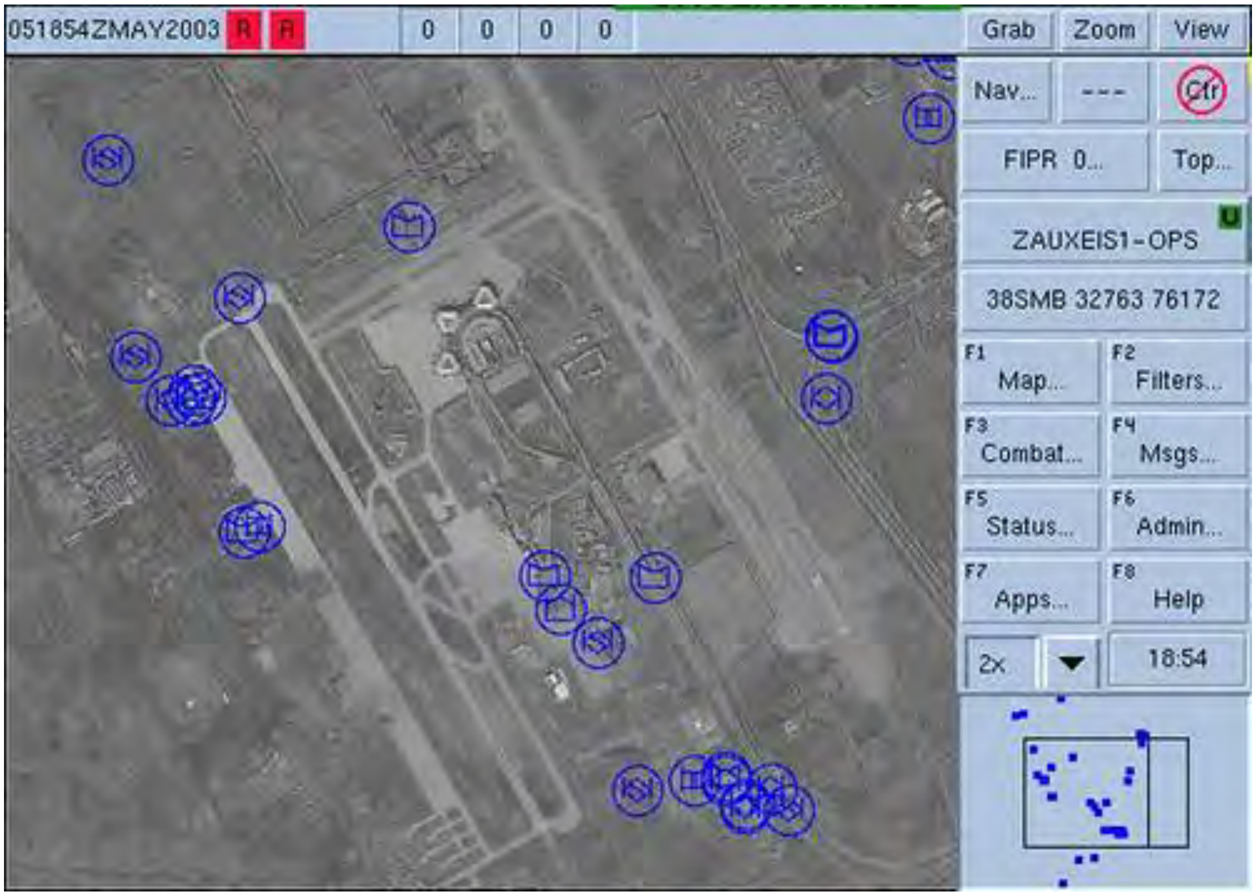
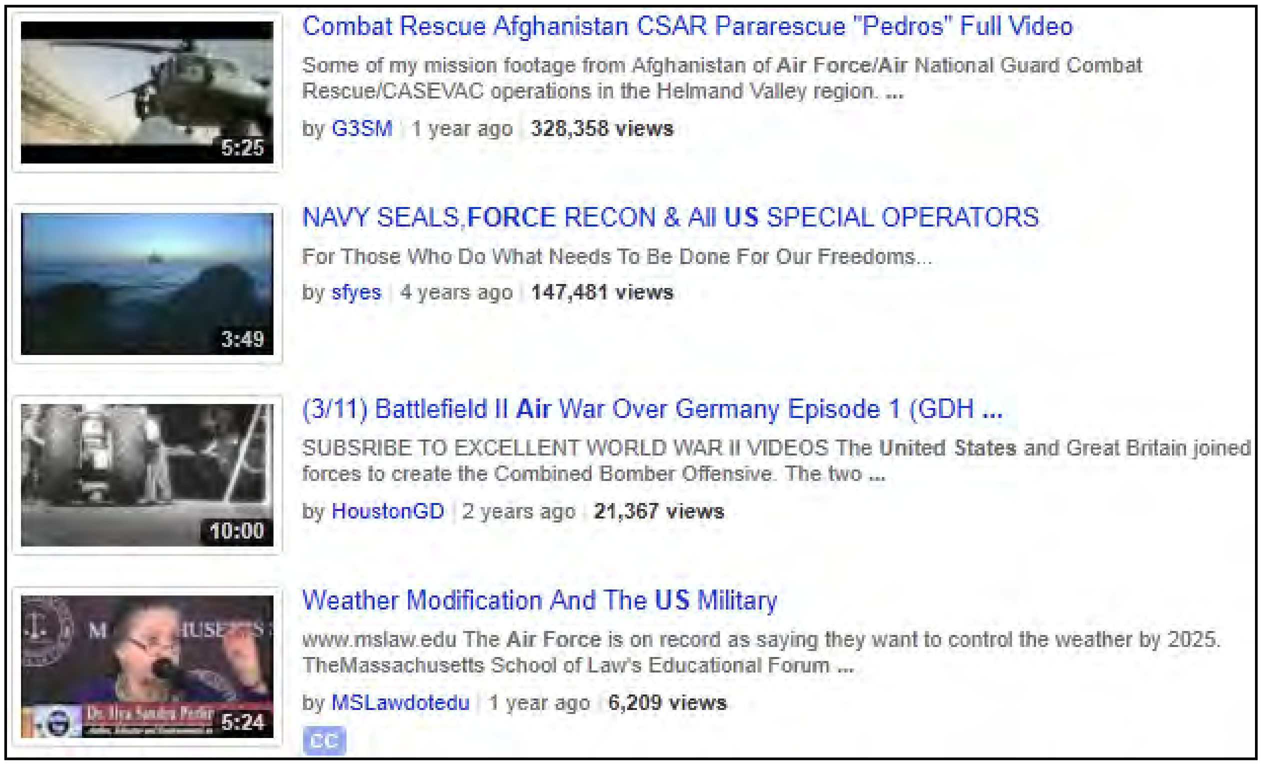


Figure 18: BFT Using Active Transponders to Track Friendly Entities

2.14.3.2 Interaction with TIPL via Queries and Orders

Tentacle Client supports live queries by providing mechanisms for filtering the display of T2DL data and for viewing live video feeds corresponding to a particular entity or group of entities. In phase II, Tentacle Client will support archived queries by providing interfaces for constructing queries of the TIPL and for displaying the results of those queries in an organized fashion (e.g., a YouTube-like interface exemplified by Figure 19), allowing the user to select a product for viewing. Upon selecting a product, Tentacle Client will then construct an order and send it to TIPL, which will return the actual video footage and associated metadata. Both live and archived products will be available to Tentacle Client, and both will be accessible on-demand to minimize bandwidth consumption.



Combat Rescue Afghanistan CSAR Pararescue "Pedros" Full Video

Some of my mission footage from Afghanistan of Air Force/Air National Guard Combat Rescue/CASEVAC operations in the Helmand Valley region. ...

by G3SM | 1 year ago | 328,358 views

NAVY SEALS,FORCE RECON & All US SPECIAL OPERATORS

For Those Who Do What Needs To Be Done For Our Freedoms...

by sfyes | 4 years ago | 147,481 views

(3/11) Battlefield II Air War Over Germany Episode 1 (GDH ...

SUBSRIBE TO EXCELLENT WORLD WAR II VIDEOS The United States and Great Britain joined forces to create the Combined Bomber Offensive. The two ...

by HoustonGD | 2 years ago | 21,367 views

Weather Modification And The US Military

www.mslaw.edu The Air Force is on record as saying they want to control the weather by 2025. TheMassachusetts School of Law's Educational Forum ...

by MSLawdotedu | 1 year ago | 6,209 views

Figure 19: Sample YouTube Search Results with Video Clip Thumbnail and Metadata

2.14.3.3 Receiving Alerts from TIPL

Tentacle Client supports an alert system, whereby Tentacle Client is automatically notified of relevant changes to the TIPL. This process/mechanism is known as an alert. For alert creation users edit an XML file. In phase II, Users will be able to create, modify, and delete alerts from within the Tentacle Client user interface.

As an example of an alert, a user might wish to be automatically notified if an entity of interest (person XYZ or a vehicle of a particular color) appears to the system, or appears within a particular geographic region. Tentacle Plugin will then send the alert construct to Tentacle Server, which will continuously monitor the TIPL and will construct asynchronous notifications to send to Tentacle Client. These asynchronous notifications will appear visually distinct to the user, thus allowing appropriate action to be taken.

2.14.4 Example Tentacle System Operational Scenario

As an example of how Tentacle System might ultimately function, consider a simple instance consisting of one Tentacle Mote, one Tentacle Client, and the requisite Tentacle Server. First, Tentacle Server is started and begins listening for connections from both Tentacle Motes and Tentacle Clients. Next, Tentacle Mote connects and begins streaming live video and metadata, which is archived by TIPL. Next, a User starts a Tentacle Client, which connects to Tentacle Server and immediately begins receiving the T2DL data feed. The user interface displays the location of each battlefield entity described by the T2DL feed, which ultimately originates from the entities being tracked by Tentacle Mote. Within the Tentacle Client user interface, these entities may appear as realistic-looking 3d avatars overlaid on high-resolution imagery in NASA WorldWind. The user may then observe battlefield happenings and drill down on interesting events. Specifically, the user may see an entity of interest, and quickly pull up the live video feed from the sensor platform performing the capture. The user may also perform a query of the TIPL to determine when and where that particular entity was last seen, and view the actual video clip of the entity at that time. This example operational scenario illustrates how Tentacle System might assist Air Force operations.

2.15 Transitioned from Google Earth to NASA WorldWind

In Tentacle Client, we transitioned from a Google Earth based GUI to one based on NASA WorldWind. Technically speaking, Tentacle Client is now a Java application that has a dependency on the NASA WorldWind Java SDK. Tentacle Client uses the WorldWind SDK's functionality for displaying a 3D map with high-resolution imagery and point/line/area overlays. However, Tentacle Client also includes network communications code to communicate with Tentacle Server, and code to manage the display, location, and appearance of entities presented in the WorldWind window.

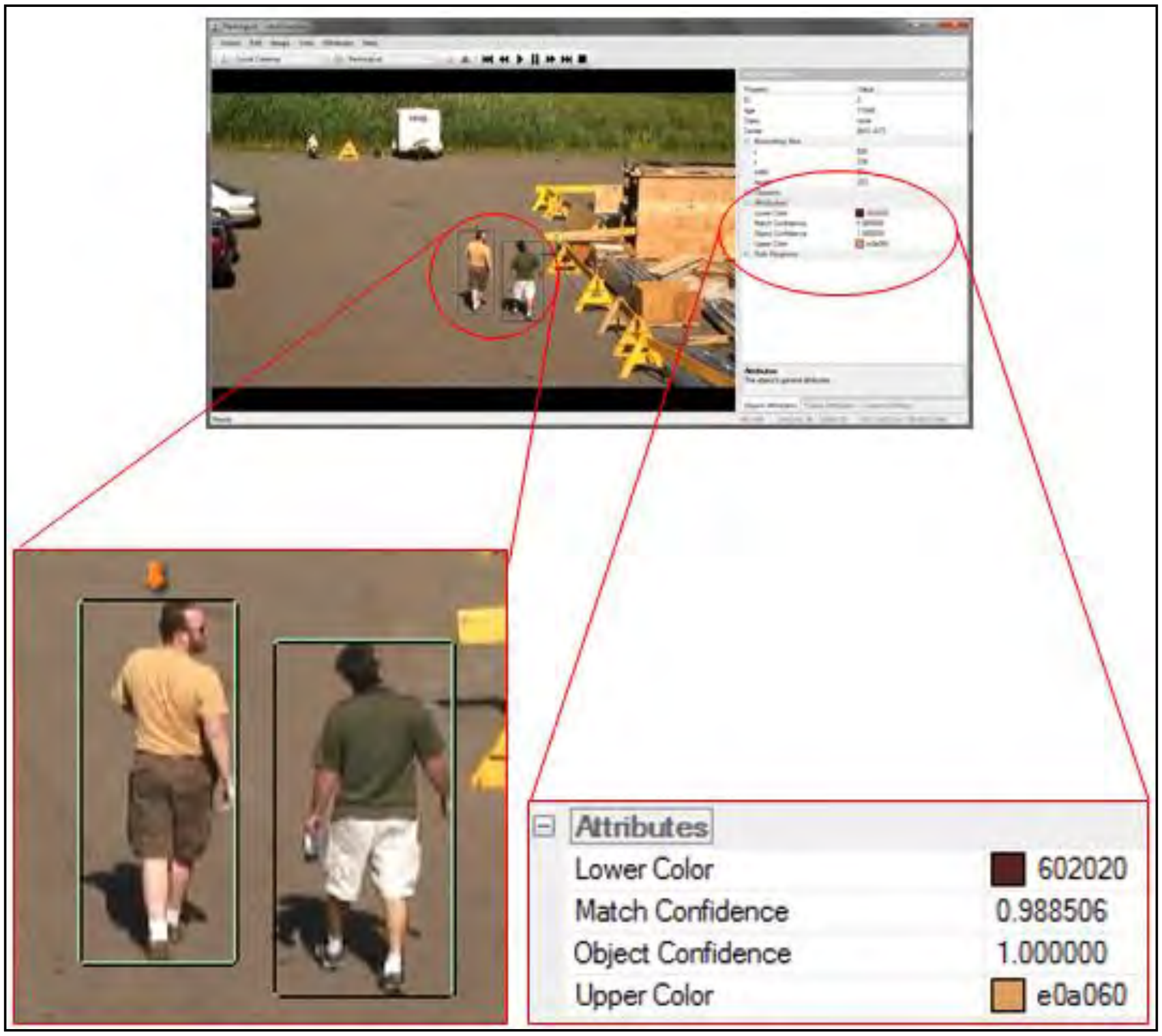
One technical note: during the migration to NASA WorldWind, we discovered that there are actually two major SDKs for NASA WorldWind, one for the Microsoft .NET framework and one for Java. Initially we integrated with the Microsoft .NET version. However, we quickly discovered that the Microsoft .NET version of the WorldWind SDK contained several glitches that would severely hamper our ability to proceed with development of Tentacle Client. In trying to get support for the Microsoft .NET version of the WorldWind SDK, we discovered that it had been largely abandoned by the open source community in favor of the Java version sometime in 2010. We found evidence of this in many ways, including more activity on Java forums and more recent commit dates for the Java SDK source code.

2.16 Upgraded to intuVision Panoptes SDK 3.7

We upgraded to a new version of the intuVision Panoptes SDK, version 3.7. For this release, intuVision incorporated:

- The ability to classify a tracked object as either a person, vehicle, or unknown
- The ability to detect the upper (shirt) color of a person being tracked
- The ability to detect the lower (pants/shorts) color of a person being tracked

This classification and color data is then packaged along with intuVisionObjects produced by the Panoptes SDK. The result is that Tentacle Mote can now publish metadata for tracked entities including entity type, upper color, lower color, position, and time. Figure 20 provides an example of how Panoptes tags the shirt and pant color of people in video as metadata.



Attributes	
Lower Color	🟥 602020
Match Confidence	0.988506
Object Confidence	1.000000
Upper Color	🟧 e0a060

Figure 20: Panoptes SDK Accurately Extracts a Person's Shirt and Pants Color as Metadata

2.17 Demonstrated Tentacle in Outdoor Environment

On 05 Aug 11, we performed our fourth internal demonstration of Tentacle, this time in an outdoor setting. A large part of testing Tentacle outdoors was setup of the Tentacle Mote to allow accurate geo-rectification of tracked entities; the remainder of the testing was aimed at simply using reproducing indoor tracking results in an outdoor setting.

2.17.1 Configuration of Tentacle Mote

Tentacle Mote produces streaming video and metadata describing entities in the video scene. The entities' positions, in geographic WGS84 coordinates, are published in the metadata stream.

Thus, a Tentacle Mote has to have a way the location of entity in 2D pixel coordinates (as seen in a frame of video) to 3D world coordinates. A necessary step for enabling this process is performing extrinsic camera calibration.

Extrinsic camera calibration consists of creating corresponding pairs of points, with one point being a 2D pixel location as measured from the top-left corner of a video frame, and the other point being a 3D world location as measured in the geographic WGS84 coordinate system. For accurate calibration, four such pairs of points are necessary.

Here is the general process describing how we performed extrinsic calibration for one of our cameras in outdoor terrain. Steps one through four of this process need only be performed once, while the remaining steps need to be performed each time the cameras are setup for outdoor use.

(1.) A person goes outside to the grassy area in front of Bandana Square on the Eastern end of the parking lot. The view is as can be seen in Figure 21.
(2.) The person stands where the camera will be and faces in the direction the camera will face
(3.) Person identifies four (4) landmarks/features that would likely be visible in satellite or aerial imagery (e.g., lines on parking lot, or a big sign) and records their descriptions in a notebook (e.g., Western-most corner of Bandana Square sign on grassy area on South side of parking lot)
(4.) Back in the office, the person uses satellite or aerial imagery to obtain the latitude and longitude coordinates of each landmark/feature
(5.) Next, the person physically sets up the camera at the location where the person stood in step one and takes care to point the camera such that all four landmarks/features are within view
(6.) Next, the person runs the ExternalCalibrator utility to record the pixel locations of each landmark/feature
(7.) The person then assembles the 2D and 3D coordinate pairs into a calibration file, which is a simple comma-delimited text file

Here is a specific example of how we calibrated one camera; first, we went to the grassy area and took several photos of the area, and then stitched them together to form a panoramic image. Using this, we can get a rough idea of where good landmarks will be, which will give a good estimate of which direction to point the camera.

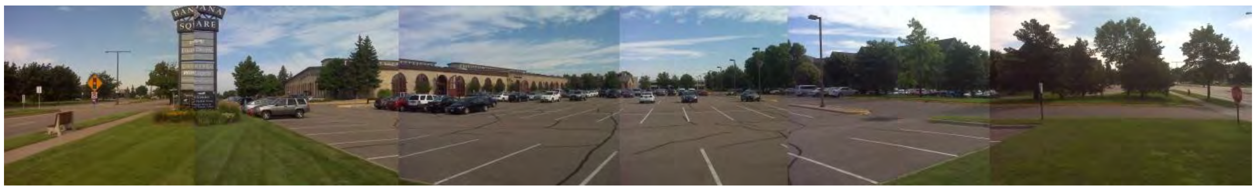


Figure 21: Panoramic Image from Camera Location in front of Bandana Square

Next, we selected four recognizable landmarks from the image, listed as follows and as depicted in Figure 22:

(1.)	Northwest corner of bench
(2.)	South corner of sign
(3.)	Near lamp post
(4.)	Far lamp post

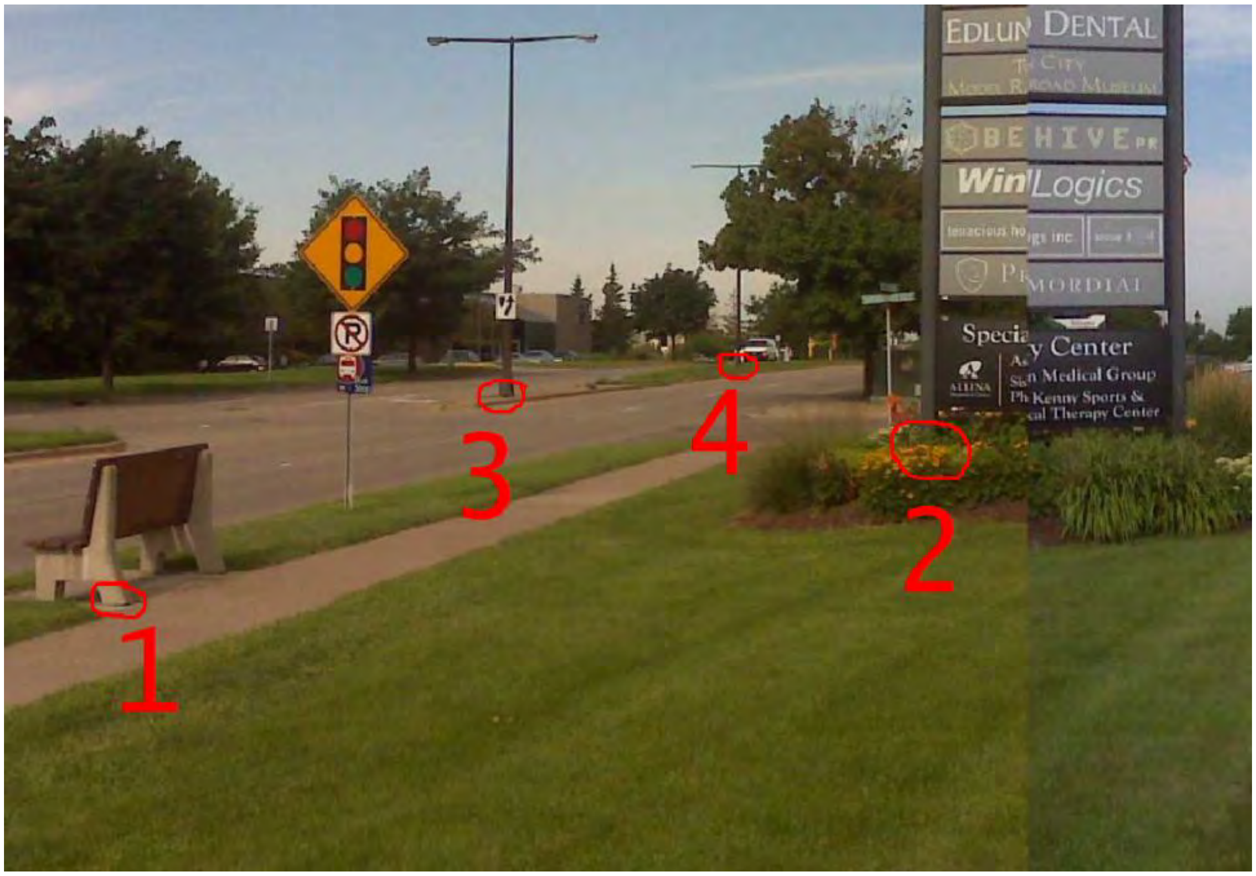


Figure 22: Landmarks Used in front of Bandana Square

Next, we inspected a high-resolution aerial image such as the one in Figure 23 to obtain the latitude and longitude of each of the landmarks:

(1.)	44.97090318° N, 93.15225008° W
(2.)	44.97094997° N, 93.15233398° W
(3.)	44.97081193° N, 93.15277076° W
(4.)	44.97079473° N, 93.15336368° W

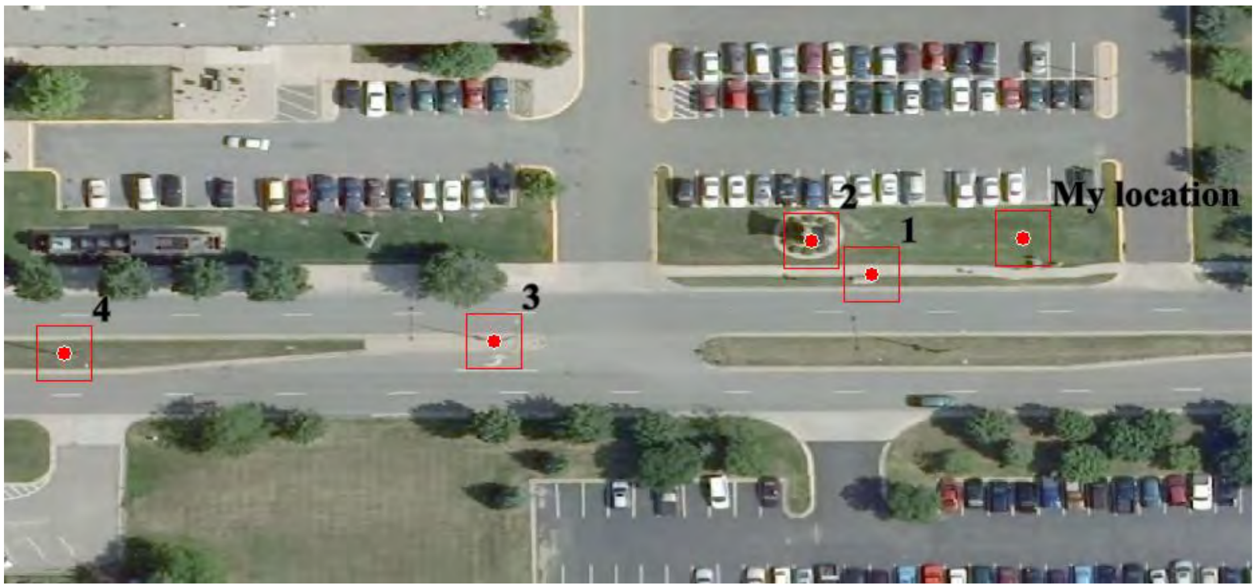


Figure 23: Aerial Image of Bandana Square with Annotated Landmarks

Then we setup a camera at our location and used our ExternalCalibrator software utility (shown in Figure 24) to identify the pixel locations of each landmark, thus allowing the 2d pixel to 3d world coordinate conversion algorithms within Tentacle Mote to take place accurately. The total equipment list for an outdoor Tentacle Mote is listed in Table 4.

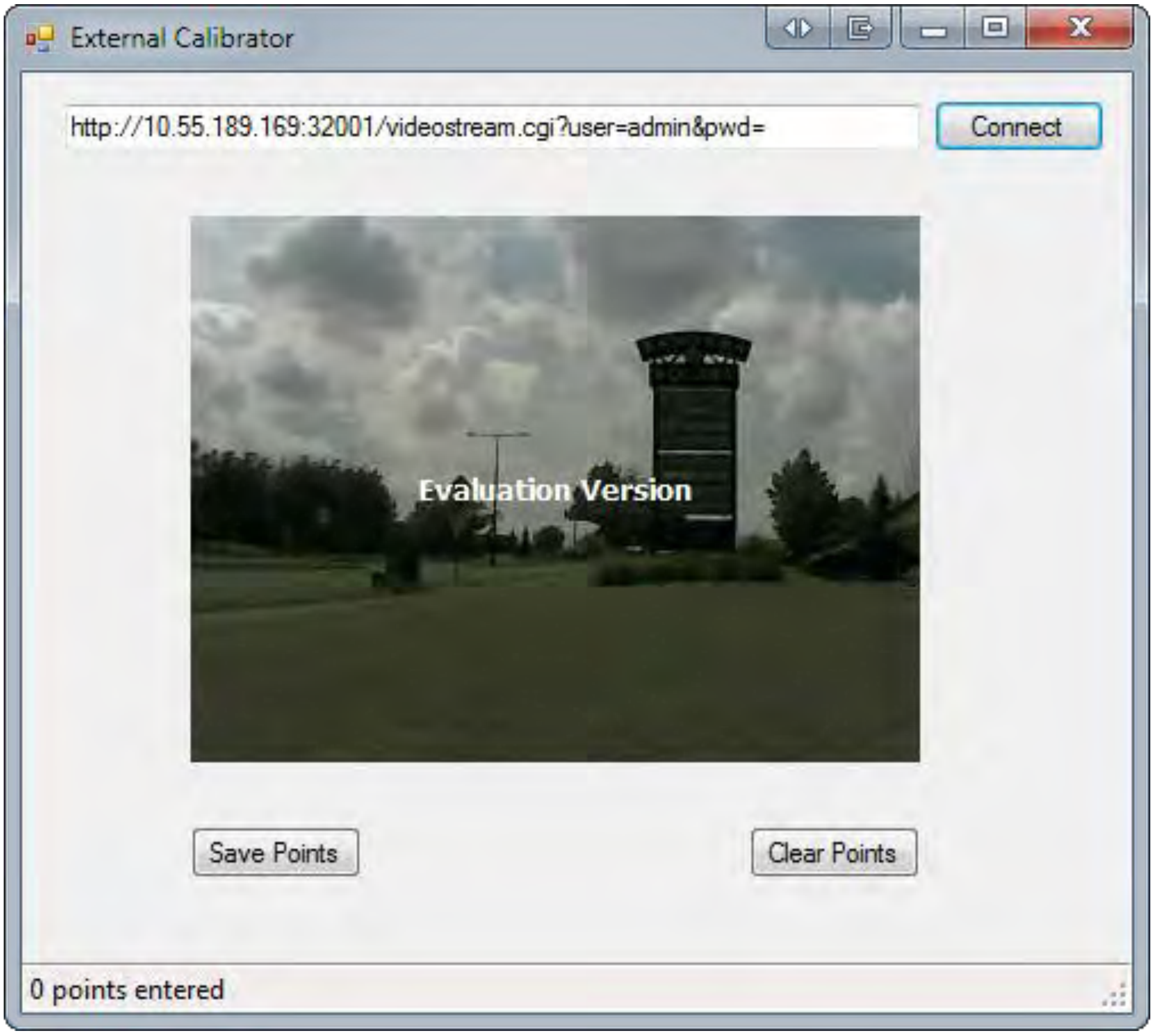


Figure 24: Software for Recording Pixel Locations of Mouse Clicks on Live Video Feeds

Table 4. Equipment List Used for Outdoor Configuration of Tentacle Mote

Item	**Quantity**
Toshiba Qosmio laptop + AC adapter + USB mouse	1
Extension cord	1
Power strip	1
D-Link DI-524 wireless router + AC adapter	1
Foscam FI8918W wireless IP camera + AC adapter	1
Folding table	1
Folding chair	2
Lab notebook and pen	1
Ethernet cable	1

2.17.2 Testing of Entity Tracking

During our testing, we identified several issues that we will need to address before external Tentacle demonstrations will become feasible:

(1.) It was necessary to adjust the parameters for object detection and classification to correctly capture both the vehicles and people in the scene

(2.) The list of entities in WorldWind grew too fast; the same object is being reported as different entities multiple times

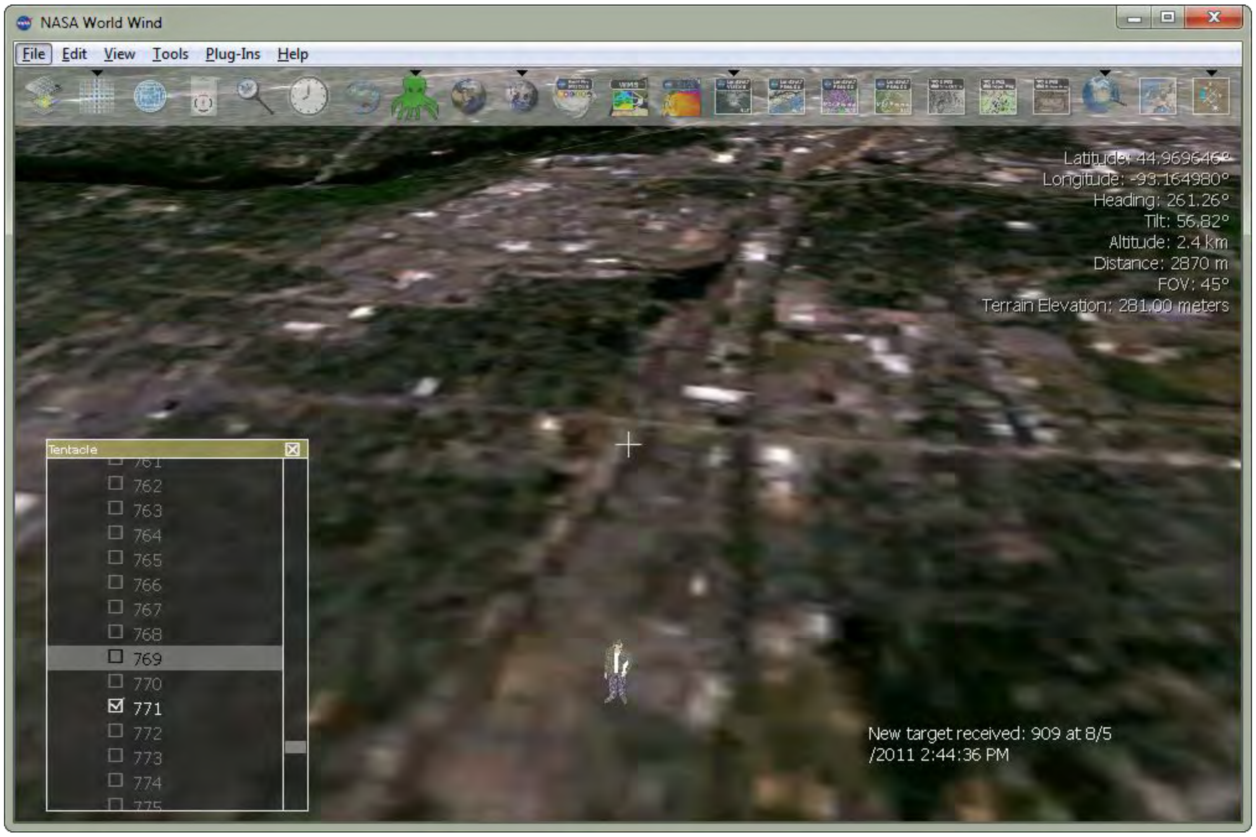
(3.) There is a bug where a dialog box appears in WorldWind stating "Unable to add new object"

(4.) There is a bug where clicking the scroll bar to browse the list of entities in WorldWind moves the cursor back to the top making it impossible to browse entities

(5.) The satellite imagery in WorldWind was of low-resolution, making it difficult to see whether or not the 3d avatar presented by WorldWind actually corresponded to the correct real-world location. See the image in Figure 25 for an example of how difficult the 3D avatar was to see.

(6.) The initial camera view in WorldWind does not automatically pan and zoom to a logical location, but does subsequently automatically pan and zoom to detected entities, leading to jerky camera movement and unpredictable user experience

(7.) The camera we used (a fairly cheap model) exhibited abrupt fluctuations in lighting, causing intuVision's object detection algorithms to go haywire since the entire background changed instantaneously. This was partly exacerbated by the fact that being in outdoor terrain subjects the camera to varied lighting conditions as clouds pass over the sun. See Figure 26 for a visual indication of the problem.



Latitude: 44.969646°
Longitude: -93.164980°
Heading: 261.26°
Tilt: 56.82°
Altitude: 2.4 km
Distance: 2870 m
FOV: 45°
Terrain Elevation: 281.00 meters

New target received: 909 at 8/5/2011 2:44:36 PM

Figure 25: 3D Avatar Overlaid on Low-Resolution Imagery

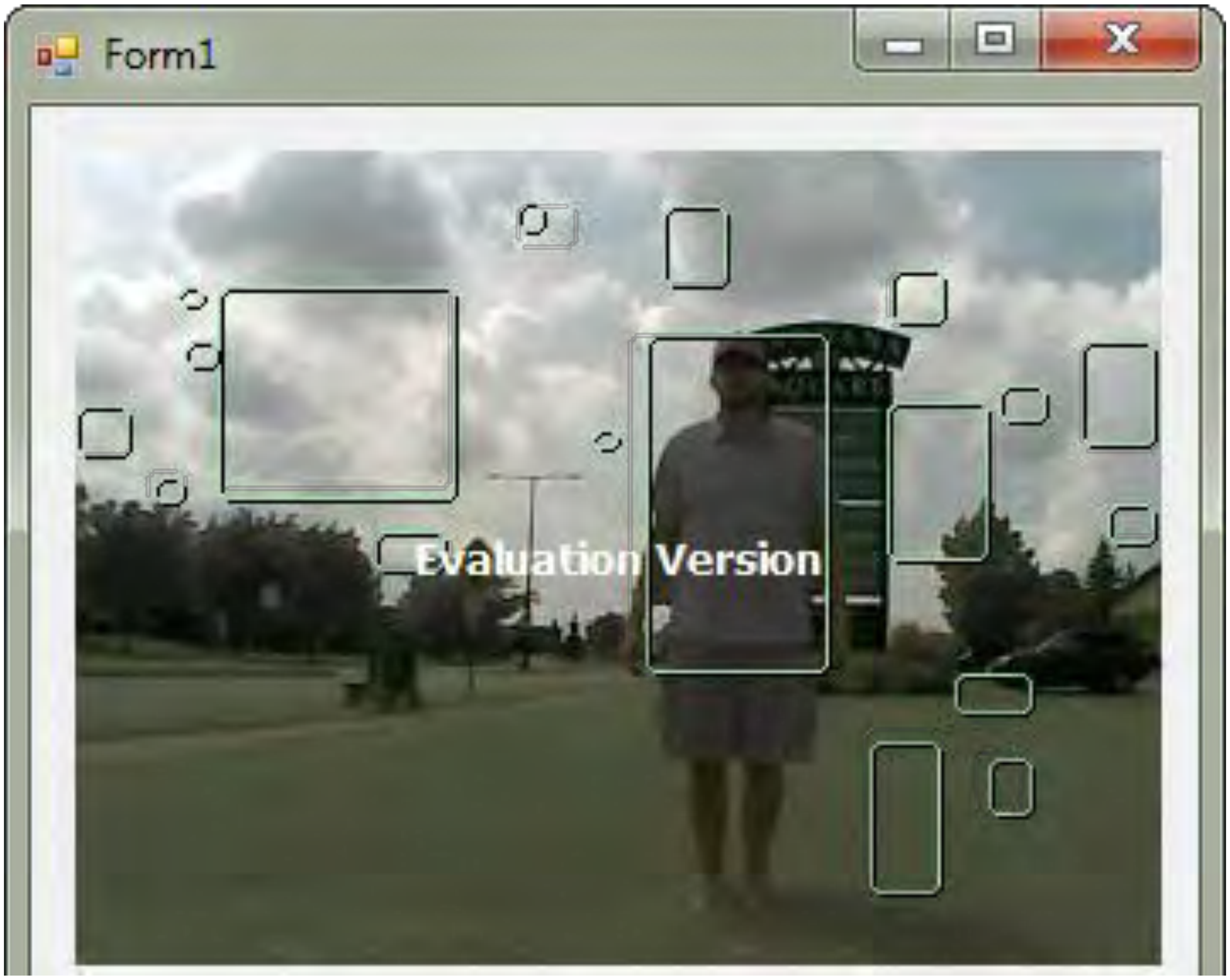


Figure 26: Artificial Abrupt Fluctuations in Lighting Make Object Tracking Difficult

2.18 Conducted Three Outings to Study Existing Systems and Customers

In an effort to ensure we are fully aware of the state of the art in video surveillance and security, we conducted three independent outings:

- Minnesota Wire and Cable
- Camp Ripley
- Atomic Data Centers

Our intent with these outings was to find out what we could about the security industry and take any relevant findings and apply them to the Tentacle project. Such findings could include discovery of a relevant user scenario in which Tentacle could provide user benefit, thus providing us leverage in phase II.

2.18.1 Minnesota Wire and Cable Entry Control System

On 08 Aug 11, we visited Minnesota Wire and Cable, a company located at 1835 Energy Park Drive, Saint Paul, Minnesota, which is less than 5 minute drive from Primordial's offices.

Minnesota Wire has installed an entry control system called SafeRise by FST21, an Israeli company. SafeRise utilizes a camera at the entrance of the building as depicted in Figure 27 to perform facial recognition and authorization of personnel against a pre-populated database of Minnesota Wire employees. The system has been used in Israel for identification of hostile persons entering the country as well as domestically as an entry control mechanism for tenants of apartment buildings.

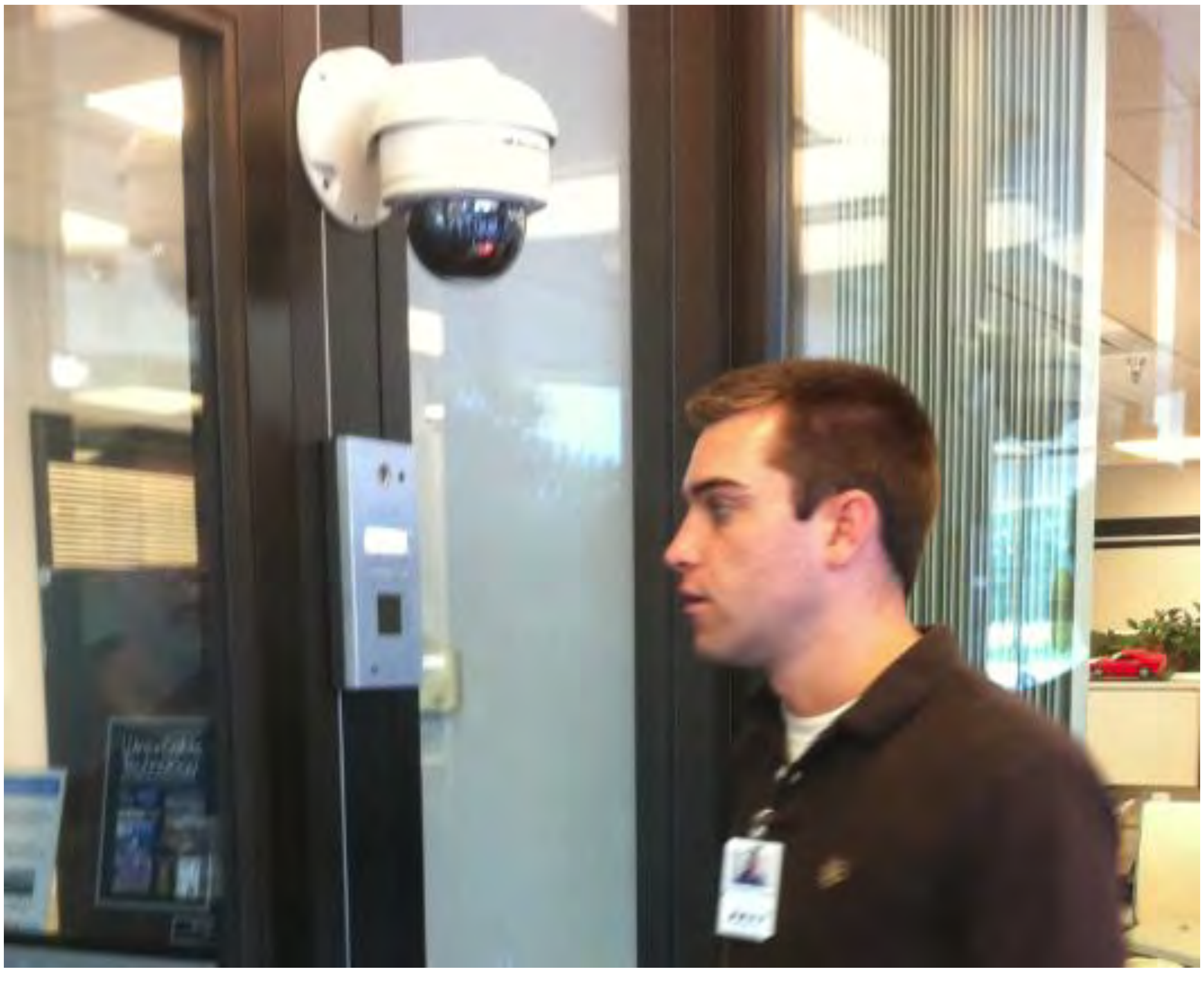
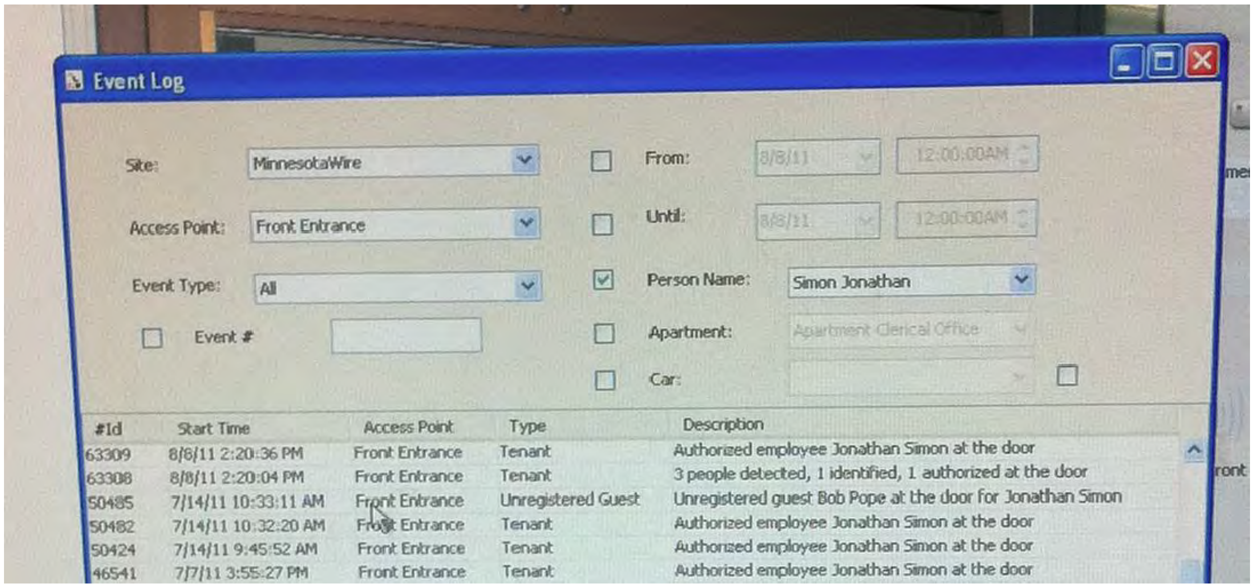


Figure 27: FST21's SafeRise Entry Control System Utilizes Cameras for Facial Recognition and Authorization

An interesting aspect of SafeRise is Event Log function as depicted in Figure 28, which allows the user to construct a query against database system events. The query can specify which camera, a date/time span, and the person name among other things. A list of results is then presented to the user, who can double click one to see still images of the person who entered the building for that event.



Site: MinnesotaWire
Access Point: Front Entrance
Event Type: All
☐ Event #
☐ From: 8/8/11 12:00:00AM
☐ Until: 8/8/11 12:00:00AM
☑ Person Name: Simon Jonathan
☐ Apartment: Apartment Clerical Office
☐ Car:

#Id	Start Time	Access Point	Type	Description
63309	8/8/11 2:20:36 PM	Front Entrance	Tenant	Authorized employee Jonathan Simon at the door
63308	8/8/11 2:20:04 PM	Front Entrance	Tenant	3 people detected, 1 identified, 1 authorized at the door
50485	7/14/11 10:33:11 AM	Front Entrance	Unregistered Guest	Unregistered guest Bob Pope at the door for Jonathan Simon
50482	7/14/11 10:32:20 AM	Front Entrance	Tenant	Authorized employee Jonathan Simon at the door
50424	7/14/11 9:45:52 AM	Front Entrance	Tenant	Authorized employee Jonathan Simon at the door
46541	7/7/11 3:55:27 PM	Front Entrance	Tenant	Authorized employee Jonathan Simon at the door

Figure 28: The Event Log Facilitates User-Constructed Queries Against All Entry Events Contained in SafeRise's Database

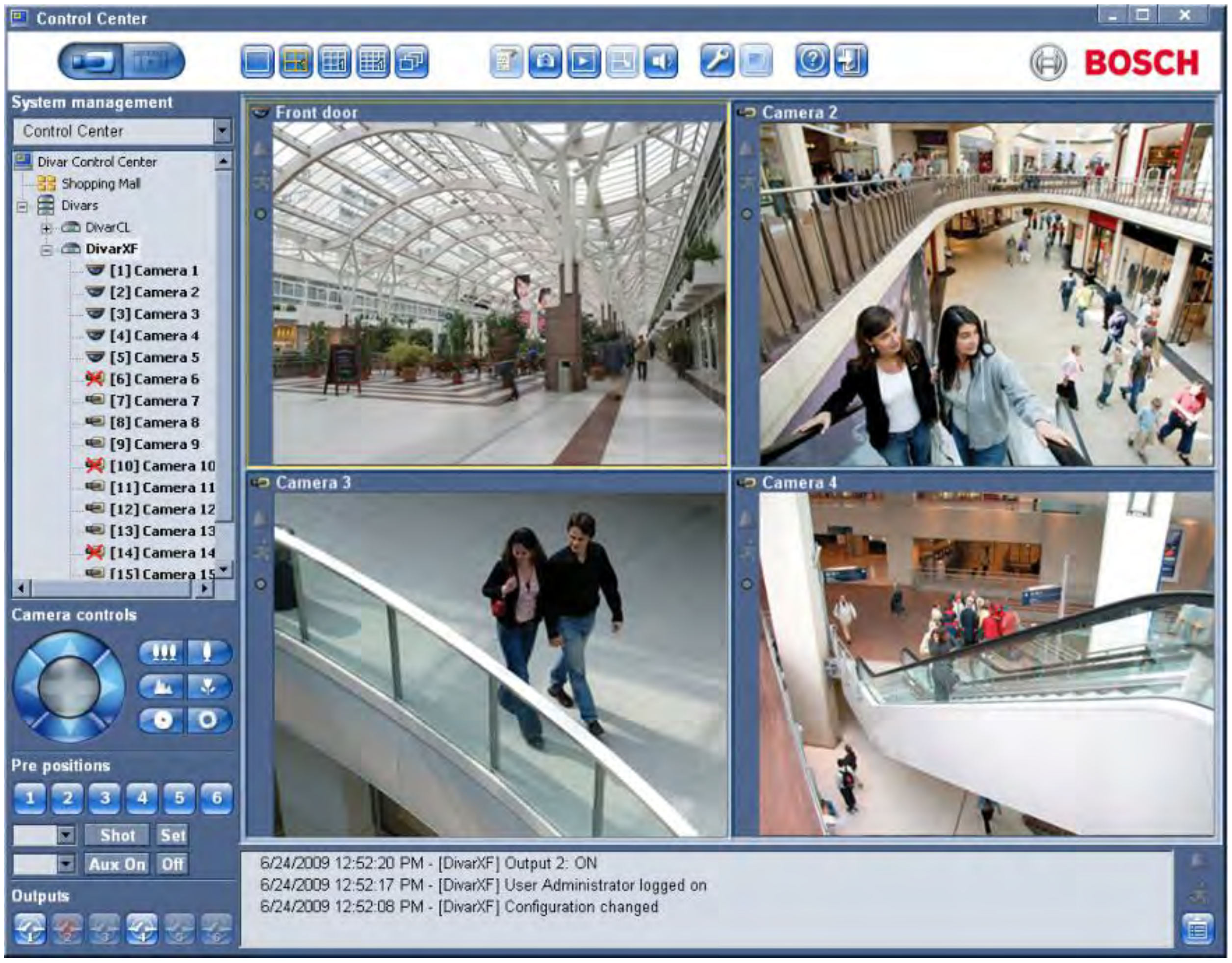
One way in which the FST21 system is not relevant for Tentacle is its reliance on a well-lit, undisturbed environment where subjects routinely stand 2 to 6 feet from the camera so that accurate facial recognition can be performed. In the case of Tentacle, we will be performing recognition of entities (people and vehicles) at a much more coarse level, e.g., by identifying primary and secondary colors. Nonetheless, we see the ability for Tentacle to persistently identify entities as a major benefit to end users, and thus plan on tackling this research issue further in phase II.

2.18.2 Camp Ripley Security System

On 15 Aug 11, we visited Camp Ripley, a U.S. Army installation located at 15000 Highway 115, Little Falls, Minnesota. We met Bruce Peterson, head of security at Camp Ripley. Our objective was to learn about how Camp Ripley conducts security operations, and specifically if and how they use a video camera system to do so.

Upon arrival at the main gate, we noticed a multiple security cameras. Later, we would learn that those cameras were for general surveillance and for license plate capture of vehicles entering and exiting the base. After arriving, Bruce brought us to a workstation where a security officer, Nick, was seated. In front of Nick was a bank of five monitors; two large TV-size monitors sitting atop a row of three smaller computer screens. Displayed on the large monitors was the Bosch Divar

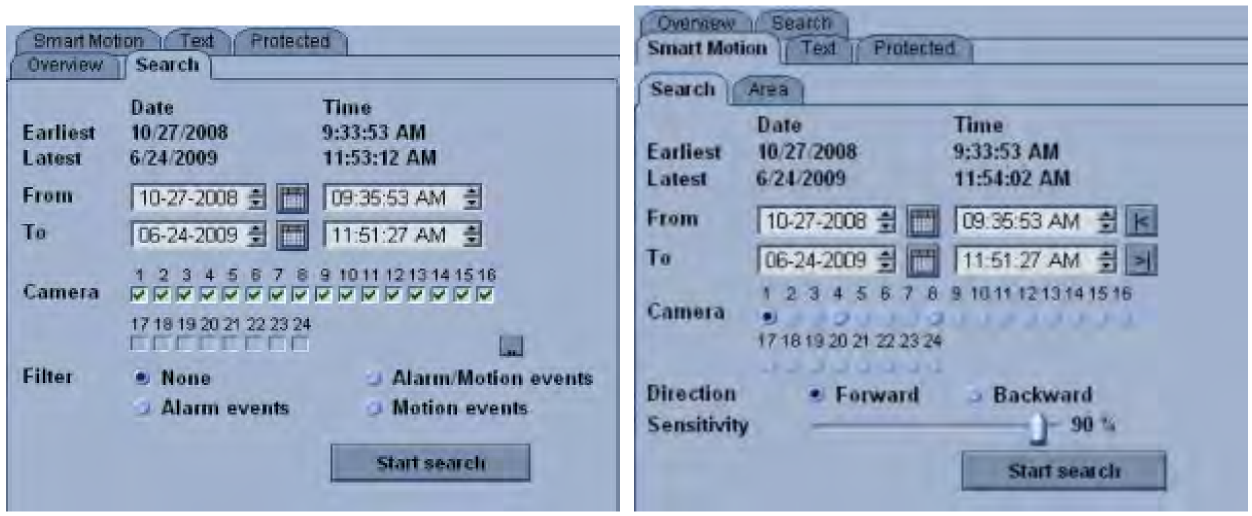
Control Center 3.1 software as depicted in Figure 29. We were not allowed to take pictures, but we were able to download the user manual for the software, which contains a screenshot:



```
6/24/2009 12:52:20 PM - [DivarXF] Output 2: ON
6/24/2009 12:52:17 PM - [DivarXF] User Administrator logged on
6/24/2009 12:52:08 PM - [DivarXF] Configuration changed
```

Figure 29: Screenshot of Bosch Divar Control Center Software from its User Manual

This software is Bosch's user interface for their closed-circuit television (CCTV) system, which is installed throughout Camp Ripley. This software is a client user interface that allows the viewing of live and archived video captured by the numerous cameras around Camp Ripley. Video is archived to multi-terabyte disk arrays. This archived video is searchable using the interfaces depicted in Figure 30, where on the left is the regular search interface, and on the right is the Smart Motion search interface, which allows filtering results based on whether motion occurs in a user-specified area of a selected camera.



	Date	Time
Earliest	10/27/2008	9:33:53 AM
Latest	6/24/2009	11:53:12 AM
From	10-27-2008	09:35:53 AM
To	06-24-2009	11:51:27 AM

Camera 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓

17 18 19 20 21 22 23 24

Filter [✓ None] Alarm/Motion events Alarm events Motion events

	Date	Time
Earliest	10/27/2008	9:33:53 AM
Latest	6/24/2009	11:54:02 AM
From	10-27-2008	09:35:53 AM
To	06-24-2009	11:51:27 AM

Camera 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 [✓ 1]

Direction [✓ Forward] Backward

Sensitivity 90 %

Figure 30: Screenshots of Archive Search Function in Bosch Divar Control Center Software

At any time, there are two or three security officers on duty; an analyst seated at the workstation, an officer at the main gate, and an officer roaming throughout post in a vehicle. During our visit, Camp Ripley had 16 cameras, some of which were indoors and others which are out of doors. In the future, Camp Ripley will have 55 cameras. The job of the security team is base security; example events include unauthorized vehicles or people entering post, sometimes during nighttime. When a security event occurs, the analyst notifies the other officers of the event via an 800 MHz radio, stating the event description (e.g., red pickup truck) and the location (e.g., main gate). The location is sometimes relayed by camera name, and each security analyst knows where each camera is. Nick has been working there for 6 years. Sometimes, when a security event occurs, and archived video must be analyzed, the analyst must first begin playing the time slice of video footage from when the event occurred (i.e. 2 pm to 4 pm on Aug 2). Each camera is available at the native frame rate of 15 FPS during playback, which can be sped up in forward or reverse to accelerate the analysis process.

2.18.3 Atomic Data Center Security System

On 18 Aug 11, we visited Atomic Data Centers, located at 615 North 3rd Street, Minneapolis, Minnesota. Our objective was to learn about how Atomic protects their computing facilities, which provide a secure environment for their clients to host websites and store critical data.

At Atomic's corporate headquarters, there is a 16-camera security system configured with cameras overlooking the two entrances to the building as well as the commons areas within Atomic's office space. The system provides live video to a bank of monitors covering a wall in front of Atomic's live support technicians. The user interface is as depicted in Figure 31. The system also writes archived video to a two-terabyte disk array, which can searched in standard ways by filtering by camera, time, or motion.

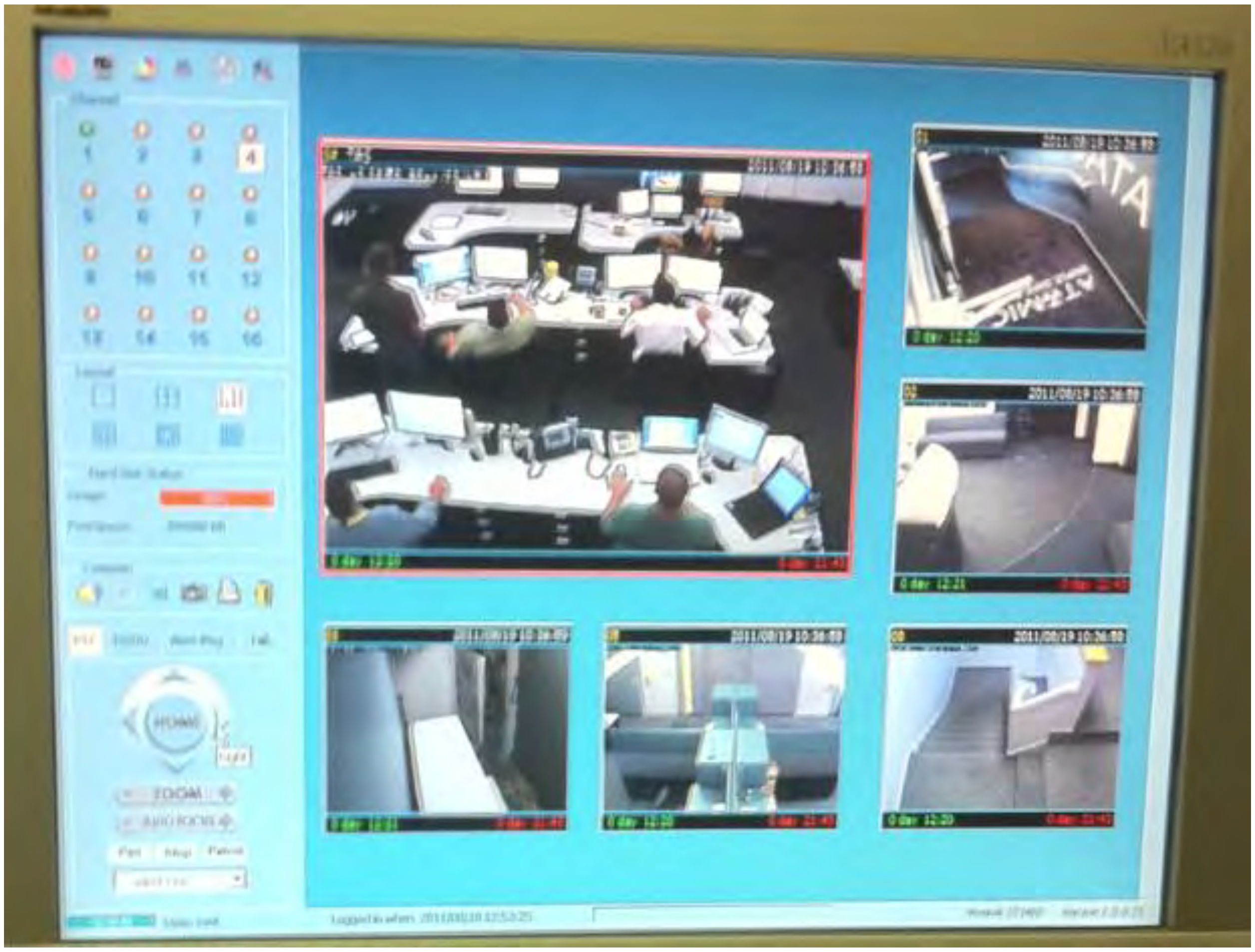


Figure 31: Atomic Data Centers Camera System

2.19 Developed In-House Tracking System

For comparison to intuVision's tracking system, as well as to mitigate risk and allow for greater flexibility, we decided to rebuild our initial in-house tracking system mentioned at the beginning stages of the project. This system is limited on overall features compared to intuVision's; however, the basic tracking element performs identically. Figure 32 below illustrates the tracking system in use.

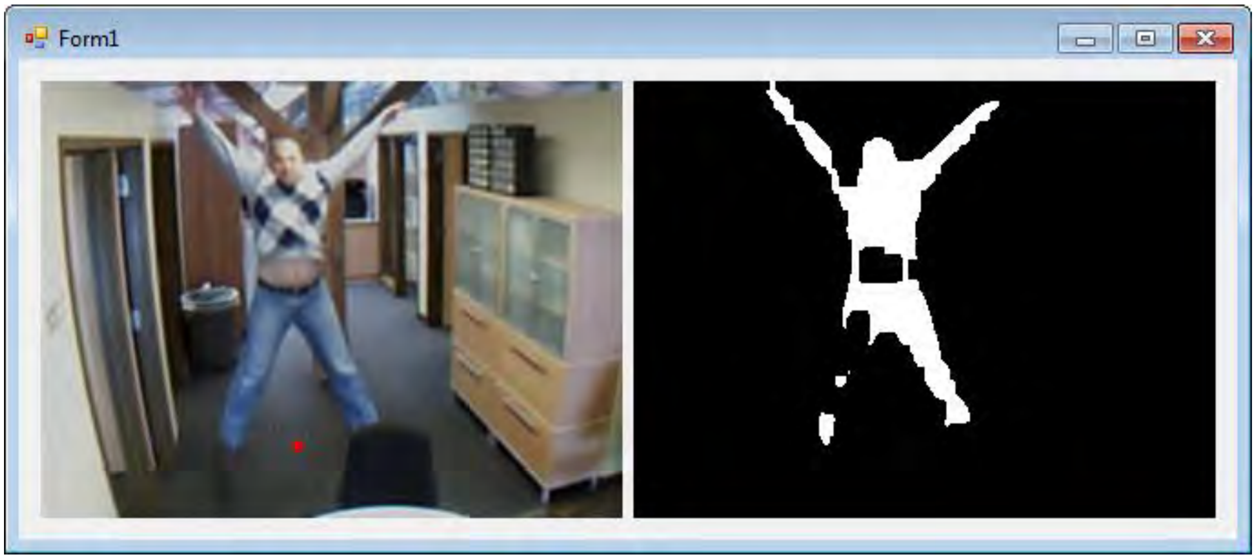


Figure 32: Primordial In-House Tracking System in Use

2.20 Improved Tentacle Client

Primordial recognized flaws with the current version of the client and made changes to rectify them. The first issue was the quality of the aerial imagery. The aerial imagery used by default in NASA WorldWind is of very low resolution and made it difficult to verify the locations of targets specified by the tracking system. We switched to a better dataset (Microsoft Bing Imagery) provided as an option in WorldWind; this improved the overall visual quality of the client as well as our ability to identify locations on the ground.

As a result of the improved imagery, it was noted that the building model location in the client was incorrect and was causing some inaccuracy in our target model positions. We adjusted the model and thus corrected the target positioning in the WorldWind view.

Finally, to improve the appearance of target paths while being tracked, we implemented a smoothing algorithm to average positions and prevent on-screen erratic movement.

2.21 Conducted End-of-Project Meeting

On 21 Oct 11 we conducted a final meeting to conclude the phase I effort. Randy Milbert and Kyle Estes from Primordial traveled to Wright-Patterson Air Force Base to partake in the meeting. Present from the Air Force were Darrel Hopper, Brian Donnelly, George Reis, Trent, and Fred. During the meeting, we discussed progress made during phase I, and we presented several video clips illustrating how Tentacle will be employed by end-users. We received positive feedback on our phase I performance, and we received initial verbal feedback on our phase I draft final report. The video clips we showed at this meeting are highlighted in Appendix A .

3.0 CONCLUSIONS AND RECOMMENDATIONS

We believe the Tentacle system succeeds in fulfilling its purpose. It tracks entities from multiple incoming sensor data feeds, categorizes data about that entity and presents it to the battlefield leader in an efficient and meaningful manner. Through exhaustive testing, Tentacle has been proven in a real-world environment. Additionally, it has been proven using low-cost, easily replaceable equipment. Although there are modifications which can be made to improve the performance and new features which can still be developed, Tentacle provides a solid framework for success in this area. We believe that Tentacle will provide a distinct advantage to leaders on the ground that are currently being overwhelmed with data.

Although the Tentacle system is currently effective, for use in the field, some modifications will be necessary and some additional features would be desirable. Below are important work items for the next phase of Tentacle development.

3.1 Support Archived Queries

One major feature missing from the current version of the Tentacle system is the ability to conduct queries on archived footage and alerts. Being able to archive video and alerts is essential for the end user as they are unlikely to be able to respond in real time to a detection warning.

3.2 Video Storage

A mechanism for storing video will need to be established which is both robust and scalable. It will need to handle numerous concurrent connections which will be adding new content as well as processing queries. It will require a significant storage capacity as well, since even a small quantity of video will consume large amounts of disk space. A database system will need to be designed to support efficient categorization of video clips and allow for complex queries on feature data. A SQL server-based system is a likely candidate. In addition, the system will need to be distributed to handle a workload of this nature and prevent data from needing to be stored in a single location. Server hardware will need to be identified as well as a plan for coordinating queries and connections between multiple servers.

3.3 Query System

Some method of querying the archived video will need to be established. It should be platform independent and scalable. This will require modifications of both the client and server systems. In the small scale, a user will need an effective interface for performing queries on archived data such as a keyword search system similar to YouTube. On the large scale, a schema must be designed for processing queries between servers. For example, an end user in one town might want to search archived data on a particular vehicle from a town further down the road. If the tracking data in the second town lies on a separate server, there will need to be a protocol for processing queries between servers as well as between the client and server.

3.4 Support Moving Cameras

Currently the Tentacle system requires static positioning of cameras to project the 3D location of targets. However, it is desirable to support cameras with dynamic positioning such as those mounted on a UAV or transport vehicle. This will require georectification of the field of view of the camera in a dynamic environment as well as constantly updating camera calibration. Additionally, the current method of feature extraction/entity detection is based on a background subtraction algorithm. The Tentacle system will need to take the camera motion into account during background subtraction in order to differentiate moving entities from moving background pixels.

3.5 Support User-Defined Rules for Alerts

In the future, the Tentacle client will be redesigned to allow the user to define rules for alerts using a simple interface. Currently these rules are manually created in a local XML file. The rules will be translated into the intuVision XML alert format and stored in the local server database. This will facilitate an audit trail and allow for easier configuration as well as allowing other users to see set alerts.

3.6 Support Open Architecture for Metadata

The Tentacle system will be modified to store all metadata in an open architecture format, namely the Cursor on Target XML schema. This will facilitate consistency across systems and allow for more efficient modification.

3.7 Improve 2D to 3D Projection

The current Tentacle system projects 2D screen points in 3D space by measuring the 3D world location where a vector projected through the 2D screen point would intersect a ground plane as defined by previously specified “tie-points”. The limitation to this approach is that all targeted points are projected at ground level, so entities on a level higher than the ground plane are incorrectly projected. In the future, this calculation will be modified to project points at a specified altitude above the ground level, which will allow for more accurate projection of entities in the view.

3.8 Support Using Custom Imagery in Client

NASA WorldWind’s default aerial imagery for projection on its virtual globe is of low-resolution. There are built-in options for slightly better imagery, provided by Microsoft Bing; however even this is of poor quality compared to most military data available. We will implement a mechanism for the user to specify custom aerial imagery in the client view to allow for more accurate tracking and visual analysis. There will be methods for the user to utilize local map data or connect to a remote map server if desired.

3.9 Identify More Effective Camera Systems

The cameras used in Tentacle were low-cost and of limited capability. This choice was intentional as to allow for deployment of Tentacle on cost-effective hardware in the long term. The system performs well in its current state; however, to eliminate errors due to light changes and shadow, as well as color recognition, it is desirable to identify better quality cameras for use in this system for more precise tracking.

3.10 Fix Avatar Display Issues

NASA WorldWind has known issues with the display of 3D models. Some models do not display textures properly; others will display in a distorted fashion or not at all. In order to properly display targets in the client application in an acceptable manner, these must be resolved. Additionally, the ability to color 3D models to match tracked targets is essential and will be implemented. As a risk mitigator, Primordial will implement feature icons with dynamic details and coloring for better entity recognition in a 3D environment, where a 3D avatar may be difficult to see.

3.11 Additional Use of Geo-Referenced Queries

In the future, Tentacle will support making queries based on location. These might include queries on tracked entities, alerts or videos based on a physical camera location (as indicated in the client window) or selected from a geographic region. An example might be that an end user may want to search for vehicles within the city of Kandahar, rather than those specifically within his local sensor network.

3.12 Support Additional Sensor Types

Currently Tentacle only supports consumption of data from camera feeds; this will be expanded to cover other sensor types being used in the battlefield. Additional sensors might be: friendly C2 systems (FBCB2 and MTS), ground vibration sensors, night and thermal imaging cameras, motion detection or audio sensors.

3.13 More Sophisticated 3D Environment

NASA WorldWind is an acceptable 3D renderer for outdoor environments but is not as capable for indoor environments. We will work with our partner All Hazards Management to facilitate development of an updated client using their C-Thru visualization application. Additionally we will mitigate risk by examining other alternatives to WorldWind for 3D visualization which might make indoor visualization more effective. This may include developing our own in-house visualization system using OpenGL or a similar 3D graphics platform.

3.14 More Efficient Human Interaction

The Tentacle client application is an initial prototype but could benefit from improvements in usability. We will leverage research in Human-Computer Interaction to improve the user interface. Areas of improvement might be to adjust the manner in which users can select or focus on entities being tracked or new methods for drawing the user's attention to a particular entity. An example might be to display flashing halos around entities in the 3D environment if they are the subject of an alert. The filtering and searching mechanisms in the interface will be improved for ease of use and efficiency as well.

3.15 Pursuer Integration

Primordial will seek to integrate the Tentacle client application into module for integration into Pursuer. This feature was omitted in phase I to allow more work on the core system functionality. For phase II, Pursuer would be an ideal platform with which to integrate the Tentacle system. We will work with the Pursuer development team to accomplish that task.

3.16 Smartphone Integration

We will attempt to use integrate smartphones into the Tentacle system, both for use as sensors and use as a client platform. In a client model, the smartphone may have a modified Tentacle client application to view camera feeds and perform entity queries and tracking. In a sensor position, the smartphone's camera could be used to track entities. Additionally, a scenario would be desirable where a user could point the phone's camera around a corner and acquire real-world coordinates for entities detected. In Phase II, we will develop a mobile application to act as a client. Primordial will also investigate the feasibility of using a mobile camera as a sensor device.

3.17 Improve Server Scalability

The current Tentacle server system is a prototype and will work only on a small scale. To cover large numbers of connections, a more robust system will need to be designed and implemented. The system will need to work in a distributed manner. Namely local queries on local data can be served locally, while a mechanism must be architected to process queries between local servers to support large scale/remote queries. Additionally, the server systems must be able to handle massive numbers of connections and queries simultaneously. Primordial will create a design for this system and demonstrate a prototype.

3.18 Work with Partners to Improve Their Contributions

Primordial will work with their partners, intuVision and All Hazards Management, to improve their product's contributions to the system:
intuVision: we will work with intuVision to improve their entity recognition/extraction (including using moving camera feeds) as well as object persistence across cameras. Currently there are issues with light changes and shadows that can throw off target detection. We will assist intuVision in making their detection more robust in these areas as well as improve our own in-house tracking system.

AHM: We will assist them in improving their C-Thru visualization application and integrating it in to the Tentacle system. C-Thru's state-of-the-art user interface will make user interaction with Tentacle Client more efficient, especially for indoor applications.

3.19 Support More Complex Behavior Pattern Recognition

We will seek to improve both our own and intuVision's tracking systems by supporting more complicated behavior pattern recognition. Currently only movement and idle behavior is supported. In the future, we would like to support detection of a larger range of suspicious behavior. Primordial will examine cutting edge research in the areas of support vector machines, neural networks, and related fields to develop a reliable method for tracking specified behavior patterns.

3.20 Real-World Testing with Military Units

Once the above features are implemented, we will seek to test with actual military units to verify and correct issues that would prevent the Tentacle system being used in a battlefield environment. Additionally we will seek to perform a demo using UAVs in an outdoor environment. This aspect is essential to prove that the system will perform as expected in a real-world scenario and can handle the burdens that are associated with it.

3.21 Urban Telepresence

Primordial will work with its contacts to see how Tentacle may fit into the Department of Homeland Security's "Urban Telepresence" project which involves entity tracking/behavior detection on a large scale in an urban environment. We feel Tentacle could make a valuable contribution to that and other similar systems and it would provide Primordial with an additional opportunity to integrate with a system being used in a real-world environment.

REFERENCES

1. **Bradski, Gary and Kaehler, Adrian.** *Learning OpenCV*. 2008.
2. **North Atlantic Treaty Organization.** STANAG 4559. [Online] http://www.nato.int/structur/ac/224/standard/4559/4559.htm.
3. **Ming, Zhao, Bu, Jiajun and Chen, Chun.** Robust Background Subtraction in HSV Color Space. [Online] http://www.mingzhao.name/publications/2002_MSA_Segmentation.pdf.

APPENDIX A - VIDEO CLIPS

Rather than demonstrate all of the features of the Tentacle system in person, we decided to create film demos which could be distributed to the Air Force and could be used for marketing purposes. Due to limitations, not all desired features could be implemented. In these cases, some features have been "mocked up" to show how they would look in the final system. The sections below show a screenshot for each demo and a description, as well as an indication of which features had been mocked up for the demonstration.

A.1 One Entity

In Figure A-1, a person walks into the office, travels the length of the office, turns and returns to the exit. This demonstrates basic tracking features. To view this video clip, open "One Entity.mp4".

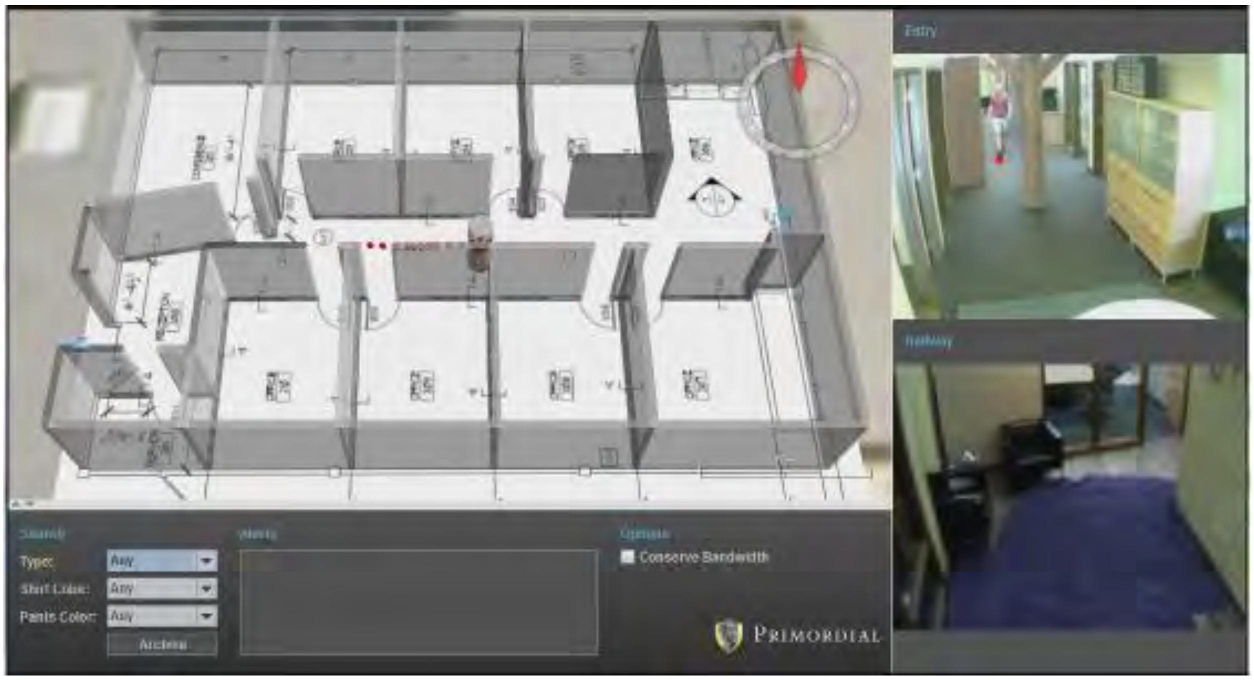
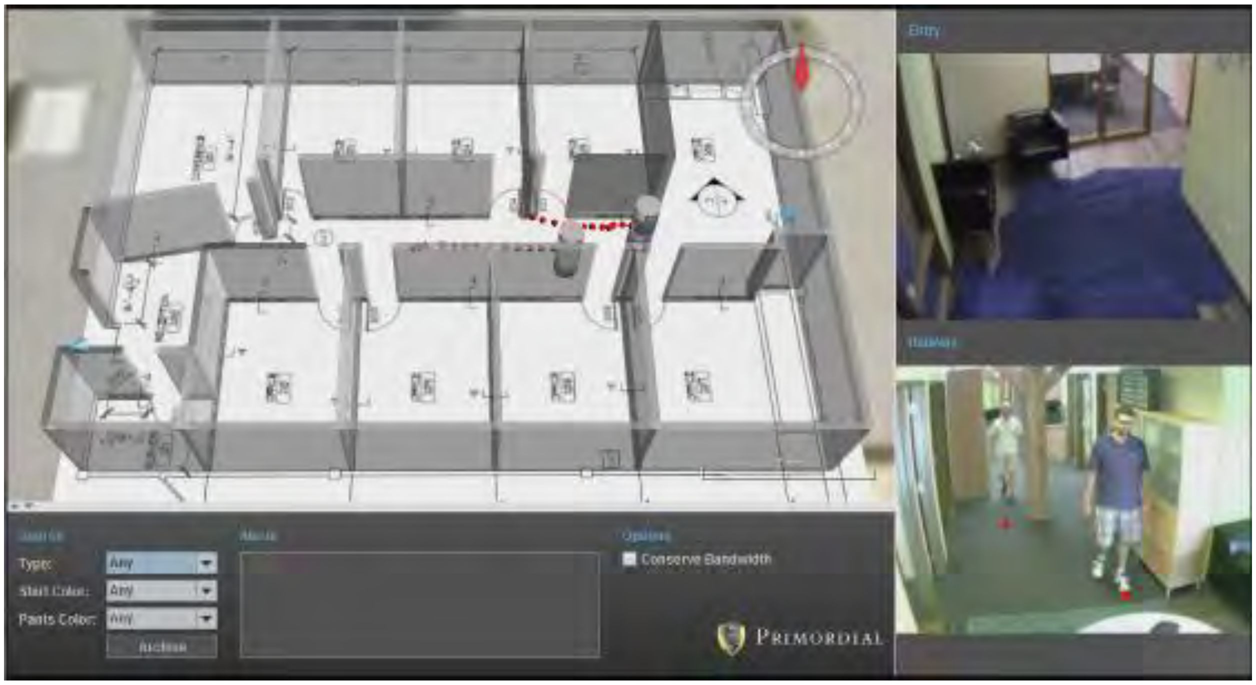


Figure A-1: Single Frame from One Entity Demonstration Video

A.2 Two Entities

In Figure A-2, two entities walk along the center office hallway and are tracked simultaneously. To create this video, it was necessary for us to hard-code the target IDs in tracking system. This demonstrates basic tracking features. To view this video clip, open "Two Entities.mp4".

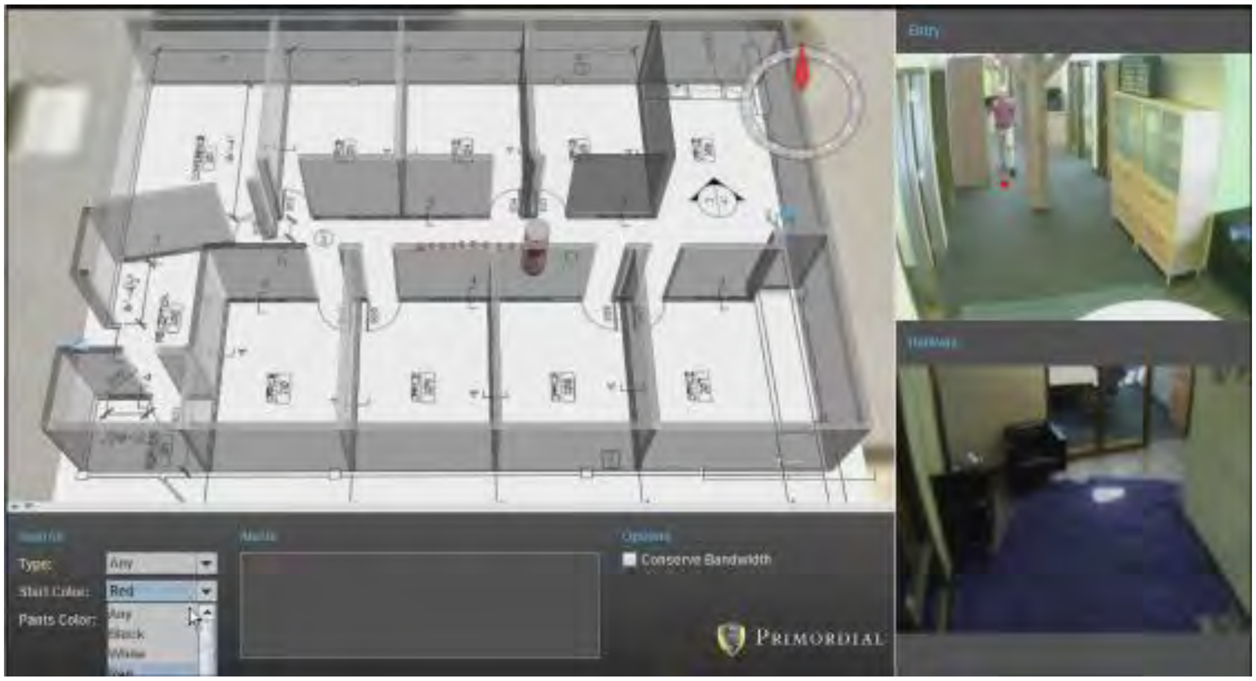


Type: Any
Shirt Color: Any
Pants Color: Any

Figure A-2: Single Frame from Two Entities Demonstration Video

A.3 Live Search

In Figure A-3, one entity is tracked, but when the user selects a live filter value that doesn't match, the entity vanishes from tracking. To create this video, it was necessary for us to hard-code color metadata for each entity due to fluctuations. This demonstrates the ability of Tentacle to filter visible entities by characteristics. To view this video clip, open "Live Search.mp4".

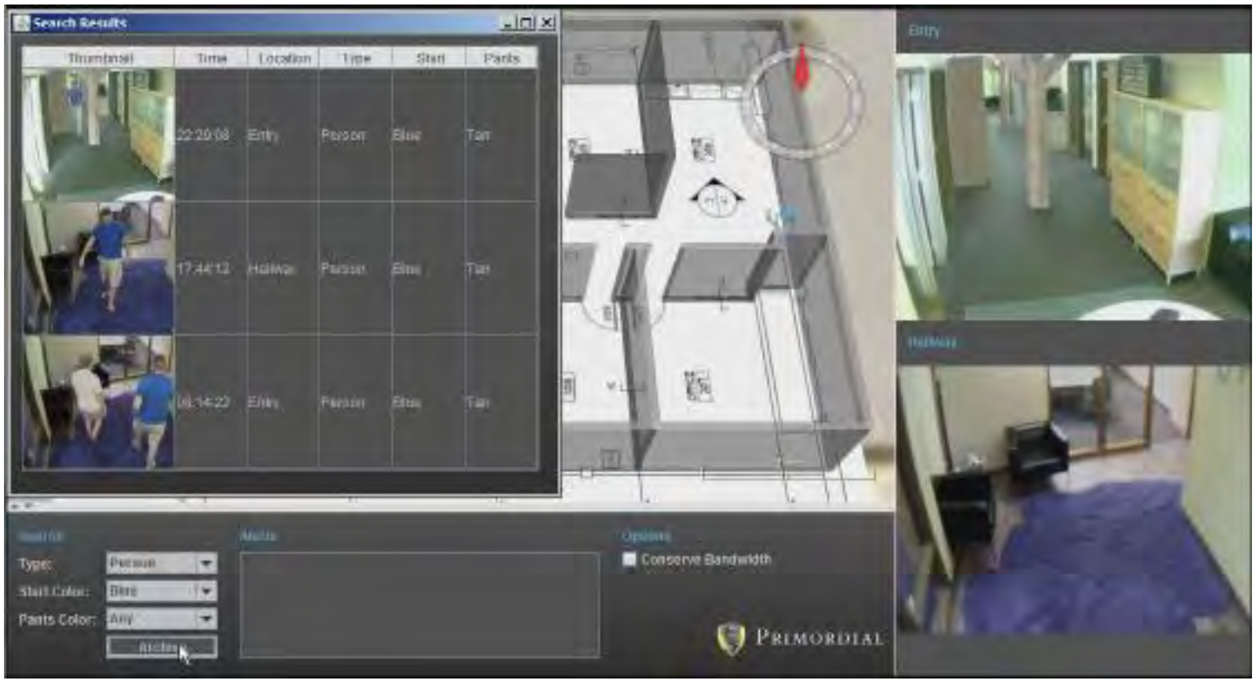


Type: Any
Shirt Color: Red
Pants Color:

Figure A-3: Single Frame from Live Search Demonstration Video

A.4 Archive Search

In Figure A-4, user selects criteria and opens archive for videos. User selects video and it plays, tracking the entity in the video in the 3D environment. To create this video, it was necessary for us to make user interface mockups for the search interface. This demonstrates how archived video search and display would be implemented in the future. To view this video clip, open "Archive Search.mp4".



Search Results

Thumbnail	Time	Location	Type	Shirt	Pants
	22:20:08	Entry	Person	Blue	Tan
	17:44:12	Hallway	Person	Blue	Tan
	08:14:22	Entry	Person	Blue	Tan

Type: Person
Shirt Color: Blue
Pants Color: Any

Figure A-4: Single Frame from Archive Search Demonstration Video

A.5 Idle Alert

In Figure A-5, a target enters, drops a bag and leaves. Alert appears, user clicks it and video plays of the incident. To create this video, it was necessary for us to mockup the interface for clicking on an alert to play a video. This demonstrates Tentacle’s ability to recognize an idle entity and flag it. To view this video clip, open “Idle Alert.mp4”.

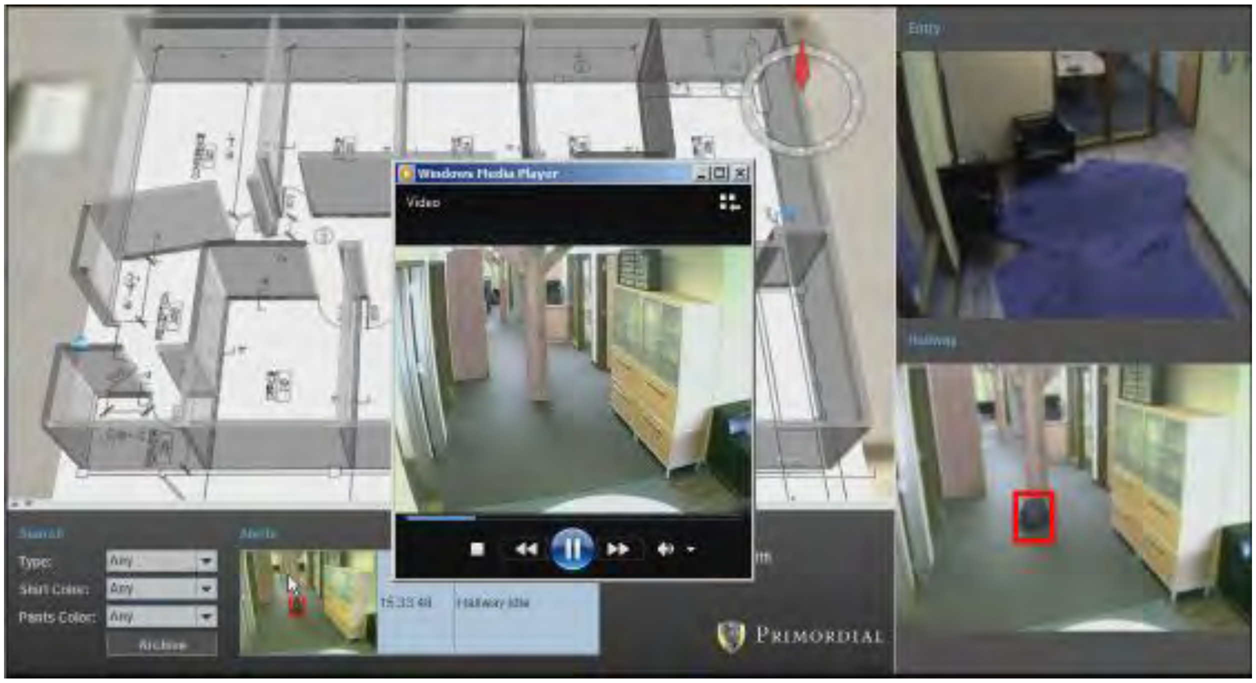


Figure A-5: Single Frame from Idle Alert Demonstration Video

A.6 Tripwire Alert

In Figure A-6, a target enters predefined “tripwire” area, alert is generated for user. This demonstrates the ability of Tentacle to flag entities entering a defined location. To view this video clip, open “Tripwire Alert.mp4”.

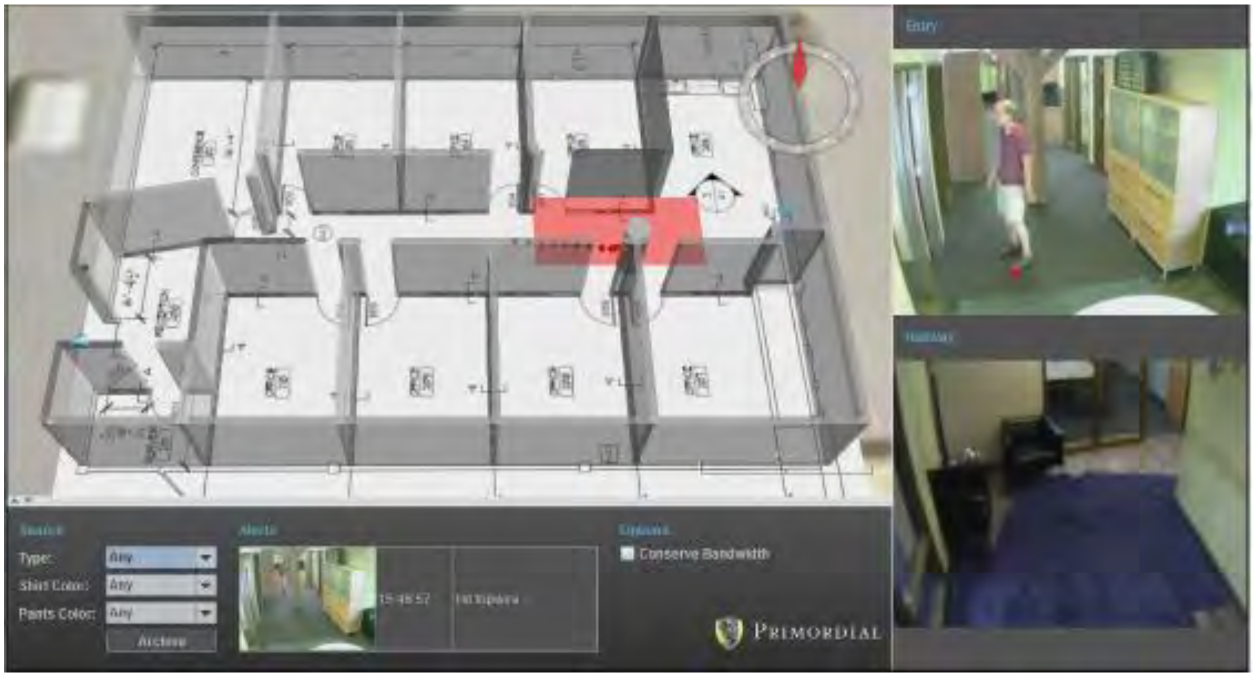


Figure A-6: Single Frame from Tripwire Alert Demonstration Video

A.7 Conserve Bandwidth

In Figure A-7, a target is being tracked, user checks “conserve bandwidth” box and video is reduced in quality. This demonstrates the lower-bandwidth functionality of Tentacle. To view this video clip, open “Conserve Bandwidth.mp4”.

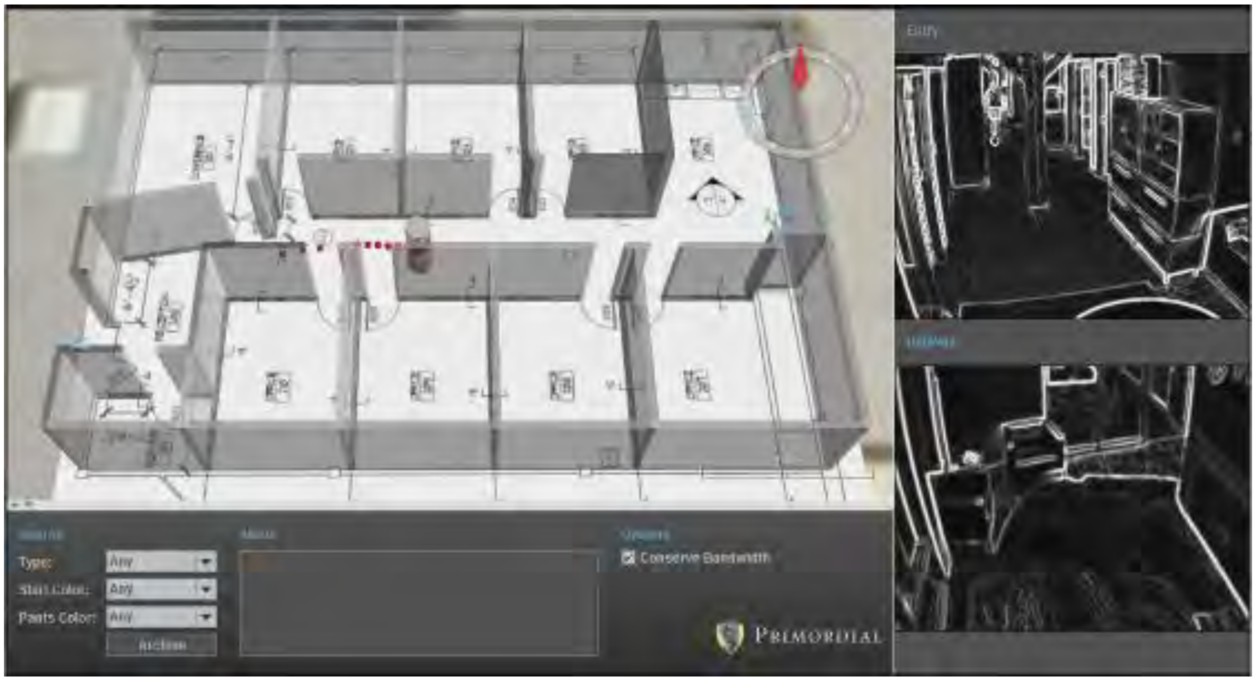

Type: Any
Start Color: Any
Pants Color: Any
☑ Conserve Bandwidth

Figure A-7: Single Frame from Conserve Bandwidth Demonstration Video

A.8 Outdoor

In Figure A-8, a person walks to vehicle, gets in, drives off camera. Person to vehicle avatar transition was assisted because tracking persistence is not yet reliable. Demonstrates how Tentacle would handle a tracking "handoff" between person and vehicle. To view this video clip, open "Outdoor.mp4".

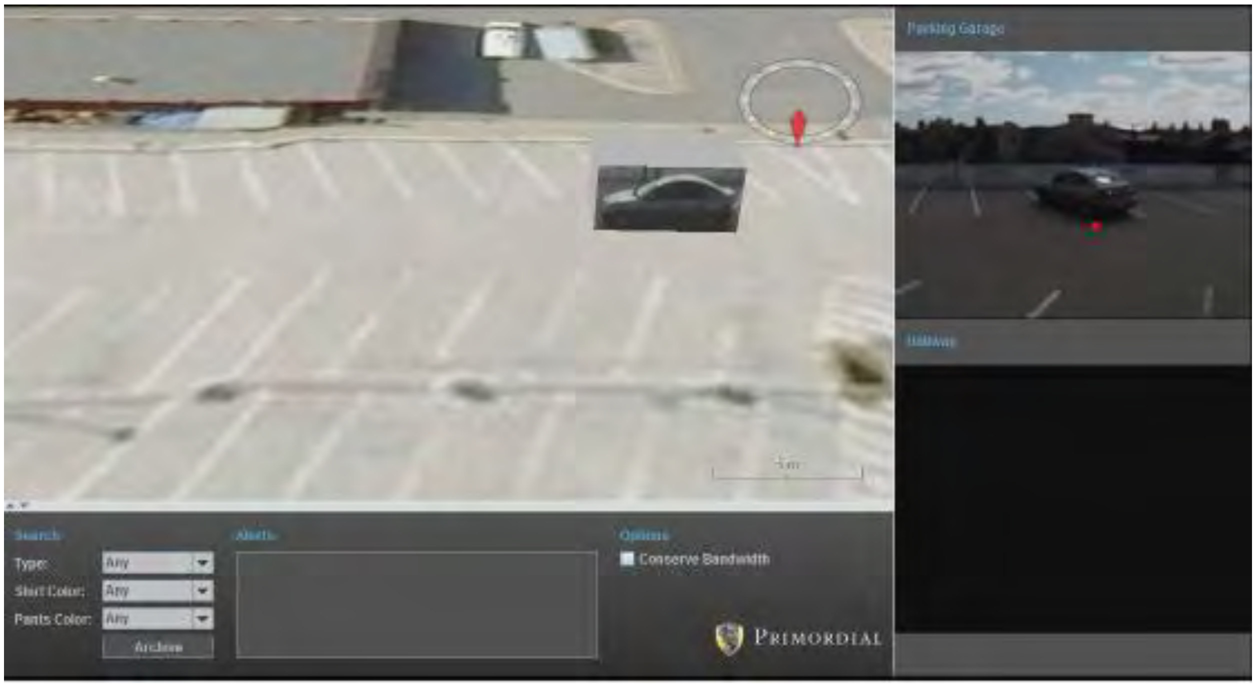


Figure A-8: Single Frame from Outdoor Demonstration Video

A.9 Downed Pilot

In Figure A-9, a scenario of multiple video feeds being tracked is shown without Tentacle then compared to a Tentacle-based approach which shows enemies converging on a downed pilot. Video is mocked up, as well as entity movement due to complexity of developing demo scenario with numerous entities. This demonstrates how Tentacle could be modified to include UAV video feeds and coordinate motes across a large area. To view this video clip, open “Downed Pilot.mp4”.

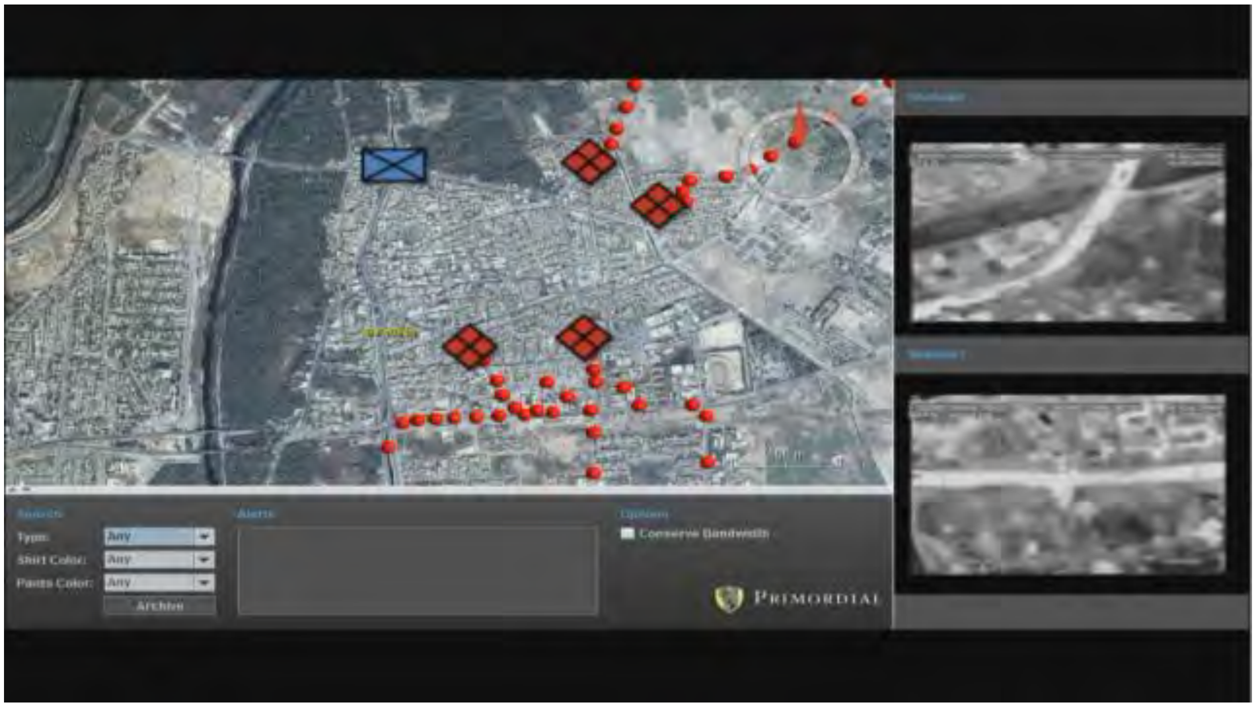


Figure A-9: Single Frame from Downed Pilot Demonstration Video

A.10 Arriving at Work

In Figure A-10, a person being tracked the entire route from parking lot to a desk in the Primordial office. Entity movement, floor map transitioning and video are all mocked up due to complexities of coordinating camera placement and light changes causing issues with the tracking system. This shows how Tentacle could be used as a whole-site security system and how multiple cameras would interact on a larger scale. To view this video clip, open “Arriving at Work.mp4”.

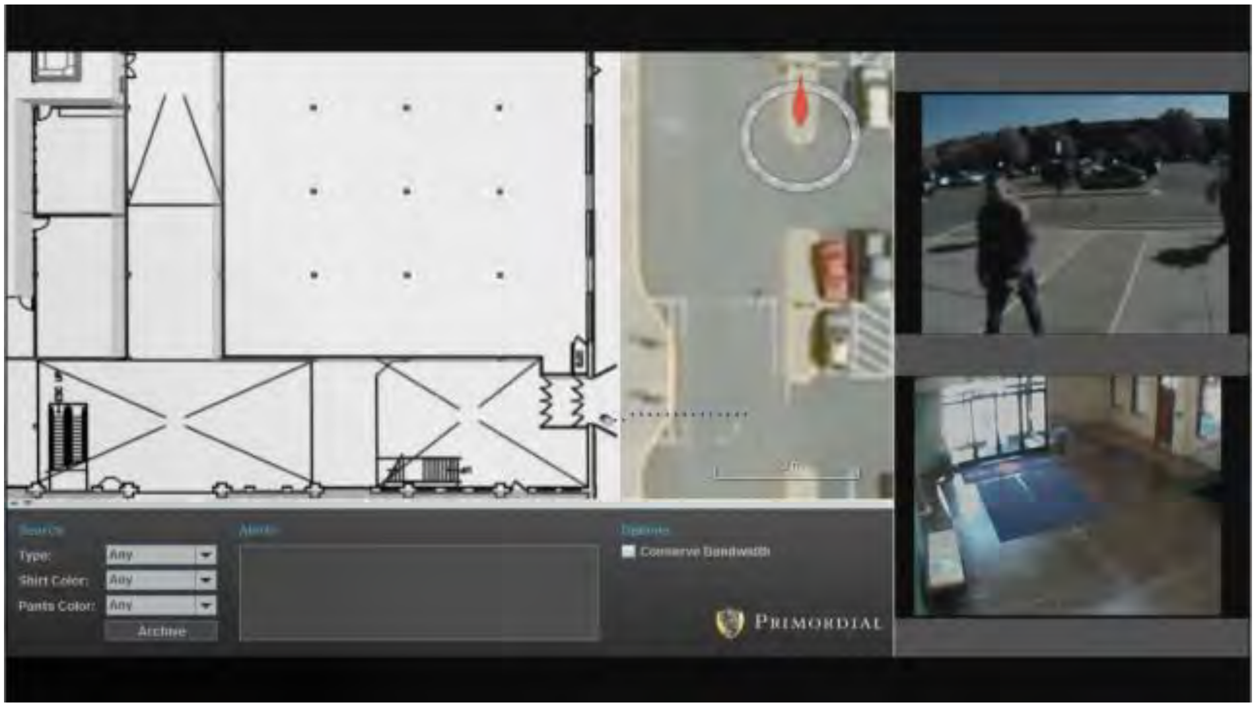


Figure A-10: Single Frame from Arriving At Work Demonstration Video